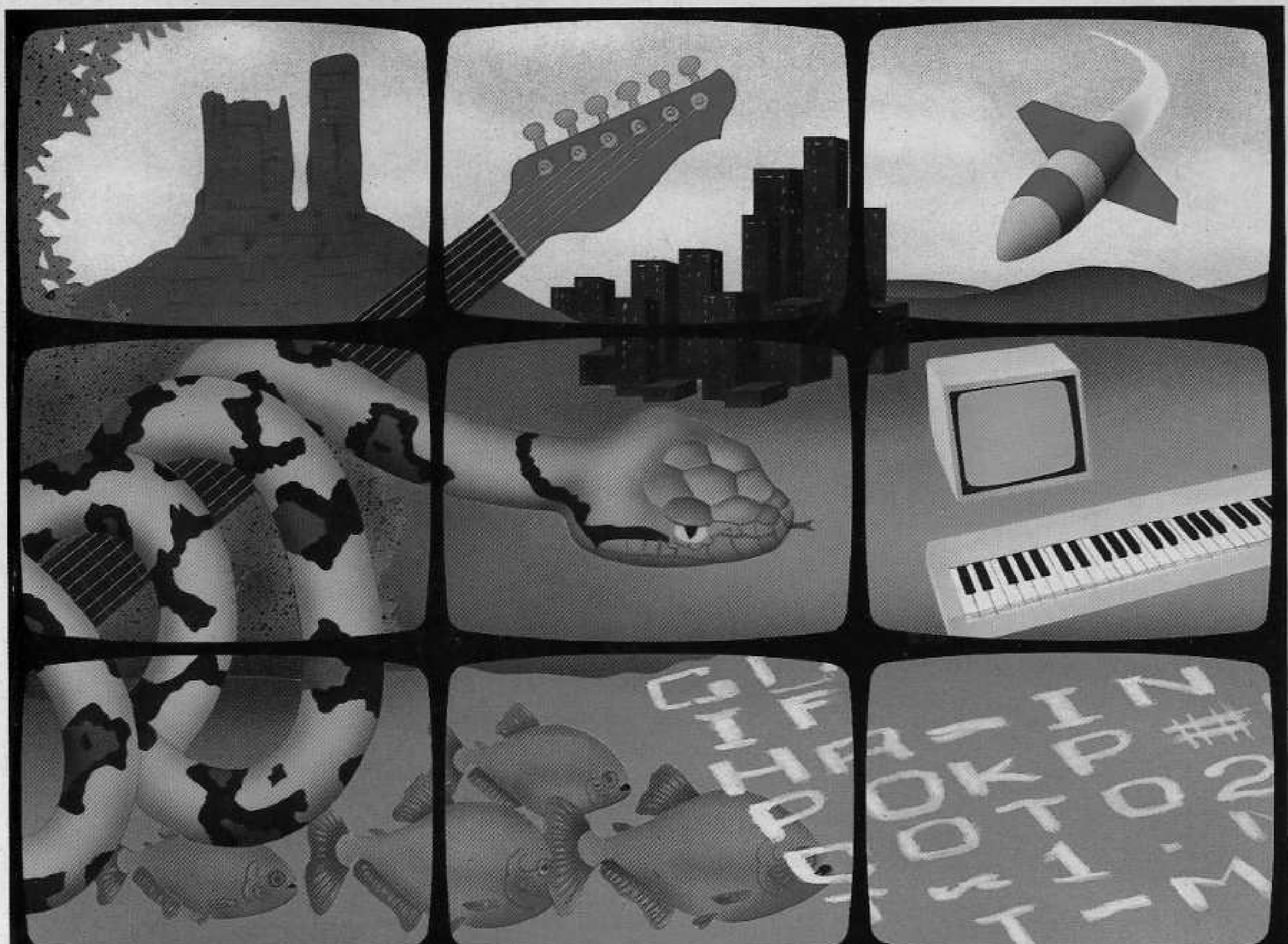
MSX

PROGRAMMES POUR MSX

Serge Pouts-Lajus, Pierre Champeaux



cedic/nathan

Programmes pour MSX

Programmes pour MSX

Serge Pouts-Lajus, Pierre Champeaux

cedic/nathan

Éditions Cedic
6-10, boulevard Jourdan - 75014 - Paris

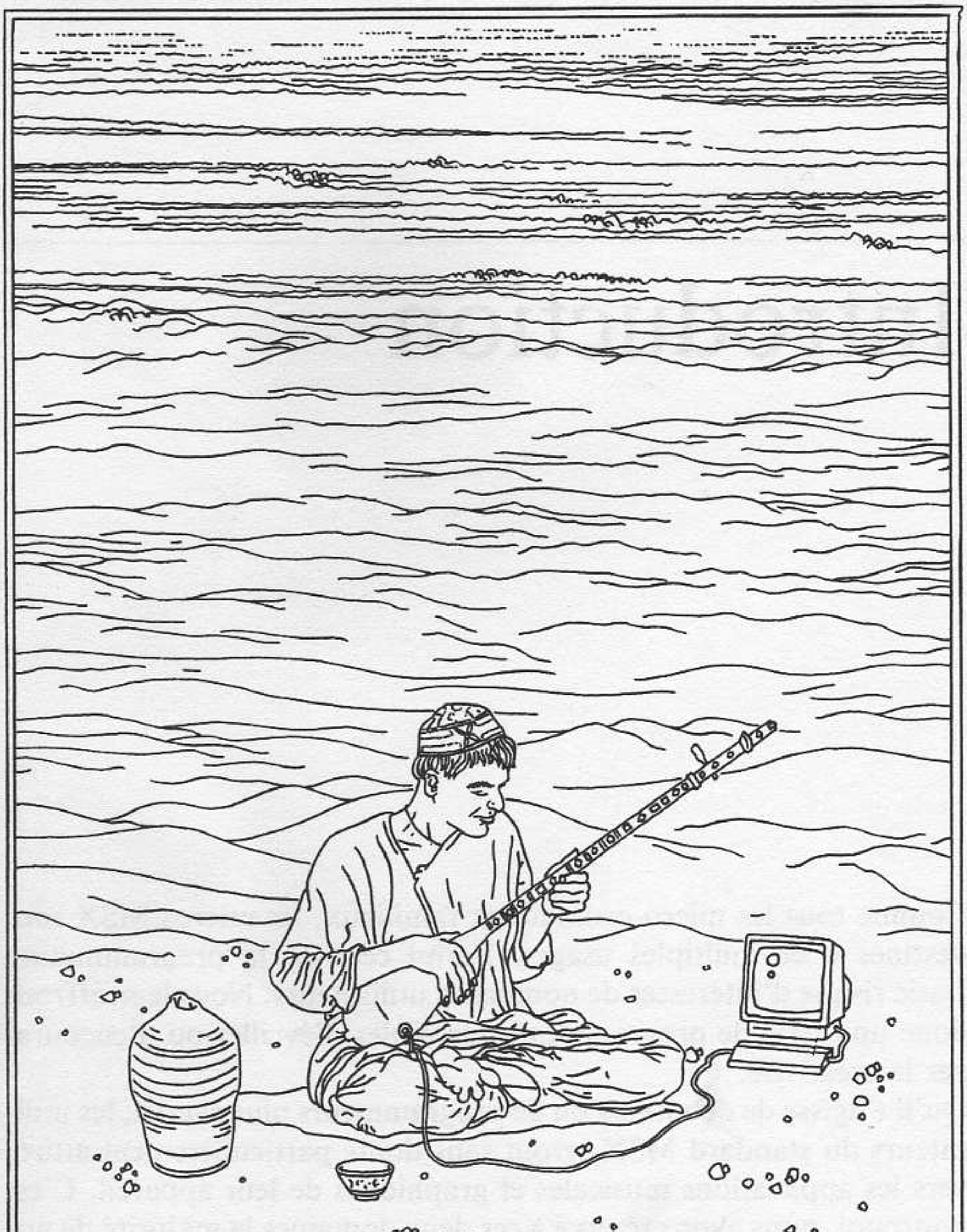
Illustration de la couverture : Daniel Brobst
Maquette : Alain Dufourcq et Jean Marc Dauvergne
Illustrations : Walter Lalonde
Composition : Rémy Couture

Ce volume porte la référence
ISBN : 2-7124-0582-X

Sommaire

Introduction

Comme tous les micro-ordinateurs familiaux, les micros MSX sont destinés à de multiples usages. Parmi ceux-ci, la programmation Basic risque d'intéresser de nombreux utilisateurs. Nous leur offrons donc une série de programmes susceptibles d'éveiller ou d'encourager la créativité.
Qu'il s'agisse de débutants ou de programmeurs plus avertis, les utilisateurs du standard MSX seront sans doute particulièrement attirés vers les applications musicales et graphiques de leur appareil. C'est pourquoi, nous avons réservé à ces deux domaines la majorité de nos programmes en mettant tout particulièrement l'accent sur le domaine sonore qui nous semble l'un des principaux atouts des micro-ordinateurs MSX.
Les listings des programmes sont suivis de commentaires qui permtront au lecteur de comprendre l'organisation et certains détails instructifs du programme et lui donneront peut-être l'envie de ne pas en rester là mais d'y ajouter un ou plusieurs grains de son propre sel.
Le dictionnaire Basic-MSX qui termine le livre recense toutes les instructions du langage et décrit leurs conditions d'emploi.


WALETERRE

Musique

Play Bach
Le derviche tourneur
Frère canon
Menuet
Random Rock
Shebam ! Blop ! Wizzz !

Les possibilités sonores du Basic MSX méritent largement qu'on s'y attarde et même que l'on commence par elles.

Pour en tirer le meilleur parti, il convient d'abord de bien connaître les fonctionnalités des deux instructions qui gèrent les registres sonores :

— PLAY
— SOUND

Nous invitons le lecteur à commencer par consulter ces deux articles dans l'annexe : Dictionnaire Basic MSX.

Play Bach

Le plus simple, le plus évident consiste à écrire un programme chargé d'interpréter une mélodie. Nous avons laissé de côté ''Au clair de la lune'' et autres fadaises pour nous attaquer à un morceau d'un niveau plus relevé.

Nous avons choisi un compositeur particulièrement apprécié des programmeurs : Jean Sebastien Bach qui eut le bon goût d'écrire des partitions extrêmement structurées en prévision de leur adaptation au micro-ordinateur. Voici donc, extrait du ''Clavecin bien tempéré'', le deuxième prélude pour pianiste et programmeur débutant.

Le morceau présente deux qualités. Toutes les notes y sont de la même longueur ce qui divise le nombre de paramètres à enregistrer par deux ; c'est la première qualité. Le morceau est très beau, c'est la deuxième qualité.

Voici d'abord le programme dans son intégralité ; des conseils pour la saisie et des commentaires suivent.

```
10 '
20 '  Jean Sebastien Bach
30 '
40 '    Praeludium II
50 '
60 '
70 CLS:PRINT "Praeludium II"
80 PRINT "J.S.Bach
90 DATA L16
100 DATA O5C,O4E-,D,E-
110 DATA C,E-,D,E-
120 DATA O5C,O4E-,D,E-
130 DATA C,E-,D,E-
140 DATA A-,F,E,F,C,F,E,F
150 DATA A-,F,E,F,C,F,E,F
160 DATA B,F,E-,F,D,F,E-,F
170 DATA B,F,E-,F,D,F,E-,F
180 DATA O5C,O4G,F,G,E-,G,F,G
190 DATA O5C,O4G,F,G,E-,G,F,G
200 DATA O5E-,O4A-,G,A-,E-,A-,G,A-
210 DATA O5E-,O4A-,G,A-,E-,A-,G,A-
220 DATA O5D,O4F+,E,F+,E,F+,E,F+
230 DATA O5D,O4F+,E,F+,E,F+,E,F+
240 DATA O5D,O4G,F+,G,D,G,F+,G
```

```
250 DATA O5D,O4G,F+,G,D,G,F+,G
260 DATA O5C,O4E,D,E,C,E,D,E
270 DATA O5C,O4E,D,E,C,E,D,E
280 DATA O5C,O4F,E,F,C,F,E,F
290 DATA O5C,O4F,E,F,C,F,E,F
300 DATA B-,F,E-,F,D,F,E-,F
310 DATA B-,F,E-,F,D,F,E-,F
320 DATA B-,G,F,G,E-,G,F,G
330 DATA B-,G,F,G,E-,G,F,G
340 DATA A-,G,F,G,E-,G,F,G
350 DATA A-,G,F,G,E-,G,F,G
360 DATA A-,D,C,D,O3B-,O4D,C,D
370 DATA A-,D,C,D,O3B-,O4D,C,D
380 DATA G,O3B-,A-,B-,O4E-,O3B-,A,B-
390 DATA G,O3B-,A-,B-,O4E-,O3B-,A,B-
400 DATA O4F,C,O3B-,O4C,O3A,O4C,O3B-,O4C
410 DATA O4F,C,O3B-,O4C,O3A,O4C,O3B-,O4C
420 DATA F,D,C,D,O3B,O4D,C,D
430 DATA F,D,C,D,O3B,O4D,C,D
440 DATA F,D,C,D,O3B,O4D,C,D
450 DATA F,D,C,D,O3B,O4D,C,D
500 DATA E-,C,O3B,O4C,O3G,O4C,O3B,O4C
510 DATA E-,C,O3B,O4C,O3G,O4C,O3B,O4C
520 DATA O3F,O4E-,D,E-,F,E-,D,E-
530 DATA O3F,O4E-,D,E-,F,E-,D,E-
540 DATA O3F+,O4C,O3B,O4C,E-,C,O3B,O4C
550 DATA O3F+,O4C,O3B,O4C,E-,C,O3B,O4C
560 DATA E-,C,O3B,O4C,O3G,O4C,O3B,O4C
570 DATA E-,C,O3B,O4C,O3G,O4C,O3B,O4C
580 DATA F+,C,O3B,O4C,O3A,O4C,O3B,O4C
590 DATA F+,C,O3B,O4C,O3A,O4C,O3B,O4C
600 DATA G,C,O3B,O4C,D,C,O3B,O4C
610 DATA G,C,O3B,O4C,D,C,O3B,O4C
620 DATA A-,C,O3B,O4C,D,C,O3B,O4C
630 DATA A-,C,O3B,O4C,D,C,O3B,O4C
640 DATA O2G,B,O3D,F,A-,F,E,F
650 DATA B,F,O4D,O3B,A-,F,E,F
660 DATA O2G,O3C,E-,G,O4C,O3G,F+,G
670 DATA O4E-,C,G,E-,C,O3A-,G,A-
700 DATA F+,C,A,F+,E-,C,O3B,O4C
710 DATA R,O5D,C,D,E-,C,O4B,O5C
720 DATA O4A,O5C,O4B,O5C,D,O4B,A,B
730 DATA G,B,A,B,O5C,O4A,G,A
740 DATA F+,A,G,A,B,G,F+,G
750 DATA D,O5G,F,G,A-,F,E-,F
760 DATA D,F,E,F,G,E,D,E
770 DATA C,E-,D,E-,F,D,C,D
780 DATA O4B,O5D,C,D,E-,C,O4B,O5C
790 DATA O4G,O5C,O4B,O5C,O4A-,O5F,E-,F8
800 DATA O4G,O5E-,D,E-,O4F,O5D,C,D
810 DATA O4E-,O5C,O4B,O5C,O4A-,F,E-,F
820 DATA G,E-,D,E-,F,D,C,D
830 DATA L32E,L64C,D,E,F,G,A-,B-,O5C,O4B-,A-,G,F16,G,E
840 DATA F32,G,F,E,F,G,A-,G
850 DATA F,E-,D,E-,F,D,E-,F
```

```
860 DATA O3B32,D,F,A-,G,F,B,F,O4D,O3,D,B,A-,G,D
870 DATA E,O4D-,O3B-,G,O4C,O3A-,F,A-
880 DATA G,B-,G,E,A-,F,B,F
890 DATA E,G,E,C,F,D,B,F
900 DATA O2C,G,O3C,D,E,G,B-,G
910 DATA A-,O4C,F,D,F,A-,O5C,O4B,O5C,O4B,F,D,L1E
999 DATA FIN
1000 READ A$
1010 IF A$<>"FIN" THEN PLAY A$:GOTO 1000
```

Commentaires

1. Conseils pour la saisie

On observera pour commencer, qu'entre les lignes 100 et 630, toutes les lignes sont doublées. Chaque ligne correspond à une mesure. Il se trouve même quatre mesures consécutives identiques (420, 430, 440, 450). Il faut donc utiliser l'éditeur pleine page pour s'épargner de la peine. Après avoir saisi une ligne et validé, on remonte le curseur, on change le numéro et on valide à nouveau.

A partir de la ligne 640, on est condamné à saisir ligne à ligne. Il est assez agréable de s'y mettre à deux : l'un dicte, l'autre saisit. La petite heure consacrée à ce travail sera largement compensée par une minute renouvelable de bonheur absolu.

2. Organisation du programme

Pour comprendre, il faut commencer par la fin. La partie active du programme est en 1000 et 1010. On lit une note A$, si ce n'est pas FIN, on la joue et on recommence à lire. Le mot FIN est mis en bout de DATA pour que le programme ne s'arrête pas sur un message d'erreur particulièrement inesthétique. FIN est en ligne 999.

Les lignes 10 à 80 servent au titre et à nettoyer l'écran pour bien profiter du morceau pendant l'exécution.

Toutes les notes du programme sont entre 90 et 910.

3. Codage des notes

— Chaque ligne de DATA correspond à une mesure.

— La ligne 90 fixe la longueur courante d'une note. 16 nous a semblé raisonnable. Pour accélérer, faire L32, pour ralentir, faire L8.

— Le codage des notes utilise la notation anglo-saxonne.

A	B	C	D	E	F	G
La	Si	Do	Ré	Mi	Fa	Sol

+ signifie dièse – signifie bémol.

L'octave est éventuellement donnée devant la note. Lorsque ce n'est pas le cas, c'est la dernière octave donnée qui reste valable.

A partir de la ligne 830 (mesures finales), la longueur des notes diminue. Si la longueur est notée derrière la note (ex. : F32 en ligne 840), c'est la longueur courante qui reste valable pour les notes suivantes (ici : 64).

Le derviche tourneur

Il s'agit d'un morceau composé par un fameux guitariste, Marcel Dadi. La technique instrumentale utilisée est celle du "picking" qui consiste à jouer la mélodie en assurant avec le pouce un accompagnement de basse régulier.

L'une des ambitions du programme consiste à faire jouer en même temps la mélodie et l'accompagnement.

Nous ignorons si le lecteur est un virtuose de la programmation mais nous n'ignorons pas que Marcel Dadi est quant à lui un virtuose de son instrument. "Le derviche tourneur" est un morceau qu'il faut jouer en accélérant progressivement. Son créateur atteint des vitesses formidables à la fin du morceau. C'est cette deuxième caractéristique qui a retenu notre attention car le Basic MSX permet assez simplement d'accélérer progressivement le tempo d'un morceau.

En plus de l'accès à deux canaux, voici donc un court programme qui vous fera découvrir de nouvelles et surprenantes possibilités.

```
1 '
2 '     Le derviche tourneur
3 '          Marcel Dadi
4 '
10 SOUND 13,10
50 A1$="L804G405C405E4R4FEDC04A4R4AG05D04B05FED04B05C4
   R404G4R4"
60 B1$="L402CE01G02E01F02F01F02F01G02D01G02D02CE01G02E"
70 A2$="04A405C4E4R4D04B05D04BG+4R4G+4G+G+A4B"
80 B2$="01A02E01E02E01E02E01E02E01E02E01E02E"
```

```
90 A3$="05C4C4C4R4"
100 B3$="01A02E01E02E"
110 A4$="04A4R4B4R4"
120 B4$="01A02E01G02D"
1000 ' Execution
1005 T%=150
1010 PLAY "T=T%;XA1$;","T=T%;XB1$;"
1020 PLAY "T=T%;XA1$;","T=T%;XB1$;"
1030 PLAY "T=T%;XA2$;","T=T%;XB2$;"
1040 PLAY "T=T%;XA3$;","T=T%;XB3$;"
1050 PLAY "T=T%;XA2$;","T=T%;XB2$;"
1060 PLAY "T=T%;XA4$;","T=T%;XB4$;"
1070 T%=T%+30:IF T%<255 THEN 1010
```

Commentaires

1. Enveloppe
L'utilisation simultanée de deux canaux oblige à prêter attention à la forme de l'enveloppe pour que le morceau sonne bien. En ligne 10, nous avons choisi une enveloppe triangulaire (10). Vous pouvez en essayer d'autres.
L'enveloppe est commune aux trois canaux.

2. Mélodie
A1$, A2$, A3$ et A4$ sont les chaînes de la mélodie.
L'interprétation doit se faire dans l'ordre :
A1$. A1$. A2$. A3$. A2$. A4$

On retrouve cet ordre dans les lignes 1010 à 1060.
Pour la transcription, nous avons utilisé la partition en prenant soin des hauteurs et des longueurs de notes ainsi que des silences (notés R).

3. Accompagnement
B1$, B2$, B3$ et B4$ sont les chaînes de l'accompagnement. Elles interviennent dans des octaves plus basses (1 et 2) que la mélodie (4 et 5). L'ordre des basses est le même que celui de la mélodie.

4. Exécution
L'exécution simultanée de la mélodie et des basses est entre les lignes 1010 et 1060. Les chaînes à jouer sont séparées par une virgule. On ne s'étonne donc pas de trouver d'abord les chaînes A puis les chaînes B avec le même numéro.

5. Accélération

Pour accélérer progressivement, on utilise la variable T %. Fixée à 150 au début (ligne 1005), elle augmente régulièrement de 30 (ligne 1070) jusqu'à atteindre 255 qui est une valeur limite pour le tempo.

La valeur du tempo doit être rappelée dans les chaînes jouées. On aurait pu l'omettre pour les lignes 1020 à 1060 mais il est facile de deviner que nous avons utilisé l'éditeur pleine page pour recopier les lignes presque semblables.

Frère canon

Il est temps de s'attaquer au 3ᵉ canal. Dans ce but, nous avons écrit ce court programme qui exécute le célèbre air de Frère Jacques en utilisant les 3 canaux et en canon s'il vous plait !

```
10 '
20 '    FRERE JACQUES
30 '
40 '
100 A$(0)="O4L4CDEC":A$(1)=A$(0)
110 A$(2)="EFGG":A$(3)=A$(2)
120 A$(4)="G8A8G8F8EC":A$(5)=A$(4)
130 A$(6)="CO3GO4CC":A$(7)=A$(6)
1000 ' Execution
1010 PLAY A$(0)
1020 PLAY A$(1),A$(0)
1030 I=0
1040 PLAY A$((I+2)MOD8),A$((I+1)MOD8),A$(I MOD8)
1050 I=I+1:GOTO 1040
```

Commentaires

1. Mélodie

Elle comporte 8 mesures que nous avons mises dans les variables de A$ (0) à A$ (7). Chaque mesure étant doublée, il suffisait de transcrire 4 et de dupliquer les 4 autres.

Cette première manœuvre est faite entre les lignes 100 et 130.

2. Exécution
Une mesure différente pour chaque canal avec exécution simultanée.

Canal 1	Canal 2	Canal 3
A$ (0)		
A$ (1)	A$ (0)	
A$ (2)	A$ (1)	A$ (0)
A$ (3)	A$ (2)	A$ (1)
⋮	⋮	⋮
A$ (7)	A$ (6)	A$ (5)
A$ (0)	A$ (7)	A$ (6)

Les lignes 1010 et 1020 permettent de commencer dans les règles : le canal 1 seul, puis le canal 1 et le canal 2. A partir de la 3e mesure, les 3 canaux sont actifs et on peut boucler sur une seule instruction.
L'opération MOD (modulo) permet d'obtenir une suite cyclique 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, etc.

Le programme s'arrête par CONTROL-STOP car l'orsqu'un canon est lancé, rien ne peut l'arrêter.

3. Suggestions
On pourra essayer de faire jouer les 3 voix à des hauteurs différentes en changeant l'octave et on pourra accélérer le tempo comme dans "Le derviche tourneur".
Cette structure peut servir pour n'importe quel canon.

Menuet

Il y a deux façons d'utiliser l'ordinateur musical.
La première est celle que nous avons illustrée dans les trois programmes précédents. Elle consiste à transcrire une mélodie existante en s'efforçant de la respecter au maximum.

La seconde, que nous allons exploiter dans le programme Menuet et dans le suivant (Random Rock) donne à l'ordinateur la possibilité de créer ses propres morceaux.

Du coup, c'est tout un univers qui s'ouvre : un seul programme génère des milliards de mélodies pour lesquelles il n'est pas nécessaire de payer de droits d'auteur. On peut même espérer en gagner si le programme produit par hasard le tube de l'année. Le Menuet n'est plus très à la mode mais qui sait, ça peut revenir.

Pour composer des mélodies aléatoires, il n'est pas intéressant de laisser trop de liberté à la machine, car dans ces conditions, on a autant de chance d'obtenir un résultat appréciable qu'en mettant un singe devant un piano. Le principe que nous avons retenu consiste à fournir une grille mélodique (des accords et des enchaînements d'accords) dans laquelle la machine tire des notes au hasard en respectant les mesures.

Notre menuet se construit autour de l'enchaînement d'accords suivant.
La mineur — Ré mineur — Sol septième — Do majeur — La mineur — Ré mineur — Mi bémol majeur.
Le programme choisit 12 notes aléatoirement dans chacun de ces accords.

```
10 '
20 '      Menuet
30 CLS:PRINT"MENUET"
40 PRINT:PRINT"Concentrez-vous un peu"
50 PRINT "et appuyez sur une touche."
60 X=RND(1):IF INKEY$="" THEN 60
70 DIM A$(7,4)
80 FOR I=1 TO 7:FOR J=1 TO 4
90 READ A$(I,J)
```

```
100 NEXT J:NEXT I
110 L%=32
120 FOR I=1 TO 7:FOR J=1 TO 12
130 X=INT(4*RND(1))+1
140 O%=3*RND(0)+3
150 N$=A$(I,X)
160 PLAY "L=L%;O=O%;XN$;"
170 NEXT J:NEXT I
180 PLAY "O3A2R8":GOTO 110
190 DATA A,E,C,A
200 DATA D,F,A,D
210 DATA G,D,F,B
220 DATA C,E,G,C
230 DATA A,E,C,A
240 DATA D,F,A,D
250 DATA E,G+,B,D
```

Commentaires

1. Données

Les 7 grilles harmoniques sont données en DATA de la ligne 190 à la ligne 250. On reconnaîtra en ligne 190, 4 notes de l'accord de La mineur et en ligne 250, 4 notes de l'accord de Mi bémol majeur.

2. Initialisation

La ligne 60 est très importante. Elle permet de simuler l'instruction RANDOMIZE non disponible en Basic MSX.

Lorsqu'on demande un nombre aléatoire RND (1), le résultat est calculé à partir d'une formule mathématique constante. Le hasard vient du fait que le calcul est itératif : chaque nouveau nombre est calculé en faisant intervenir le précédent. Mais si vous écrivez un programme qui imprime 10 nombres aléatoires, à chaque fois que vous ferez RUN, vous obtiendrez les 10 mêmes nombres.

Pour éviter cet inconvénient, on demande à l'utilisateur de taper une touche quelconque (message en 40 et 50). La ligne 60 attend la frappe de cette touche, elle boucle tant que INKEY$ (le buffer clavier) est vide. A chaque passage on calcule un nombre aléatoire. C'est l'utilisateur lui-même qui sans le savoir, initialise la suite des RND d'une façon aléatoire.

Cette procédure doit être utilisée à chaque fois que l'on désire qu'un programme se déroule d'une façon véritablement variée. Ici, c'est essentiel si l'on veut que le Menuet obtenu soit différent à chaque fois que le programme est lancé.

3. Exécution

La partie essentielle du programme est dans les deux boucles imbriquées entre les lignes 120 et 180.

La boucle en I correspond aux 7 mesures. Celle en J aux 12 notes de chaque mesure.

X permet de choisir l'une des 4 notes de l'accord.

0% permet de choisir une octave.

Le programme s'interrompt par CONTROL-STOP.

4. Suggestions

On peut facilement changer les notes des accords pour obtenir un autre genre musical.

On peut aussi envisager de remplacer la notation littérale des notes par une notation décimale (voir article PLAY dans le dictionnaire) qui permettra des transpositions. Ajouter ou retrancher 1 pour monter ou descendre d'un demi-ton.

Random Rock

60 MILLIARDS DE ROCK AND ROLL !

Les vrais Rockers n'en reviendront pas. Leur banane en tremblera de surprise !

Voilà le rock du hasard, mitonné pour eux par un micro-ordinateur. Du rock comme s'il en pleuvait. Plus qu'il n'en faut à toute une vie de rocker. Sont-il tous bons ? Bien sûr que oui !

Combien de tubes parmi eux ? Sûrement plusieurs centaines. Encore faut-il avoir la chance de tomber dessus. Ce que nous vous souhaitons - Yeaah !

Le principe de construction de ce programme est assez semblable à celui du précédent (Menuet) mais les contraintes inhérentes au genre musical concerné obligent à raffiner la méthode. Comme le sujet s'y prête, nous avons ajouté sur le deuxième canal, une ligne continue de basse d'accompagnement.

```
10 '
20 ' Random rock
30 '
40 '
50 '
60 DIM NN(12)
70 GOSUB 420
80 RESTORE 300
90 FOR I=1 TO 3:FOR J=1 TO 4
100 READ ACC$(I,J)
110 NEXT J:NEXT I
120 FOR I=1 TO 6
130 READ B$(I)
140 NEXT I
150 RESTORE 360
160 FOR I=1 TO 12
170 READ NN(I)
180 NEXT I
190 FOR I=1 TO 8
200 OCT%(I)=4+INT(RND(1)*3)
210 HN(I)=1+INT(RND(1)*4)
220 NEXT I
230 PLAY "L64","L64"
240 FOR I=1 TO 12:FOR Z=1 TO 8
250 N=NN(I):A$=ACC$(N,HN(Z))
260 IF Z MOD 4=1 THEN B$=B$(1+INT(Z/4)+2*(N-1))
    ELSE B$="R64"
270 GOSUB 370
280 NEXT Z:NEXT I
290 GOTO 230
300 DATA C,E,A+,G
310 DATA F,A,C,D+
320 DATA G,B,D,F
330 DATA O2C,O3C
340 DATA O2F,O3F
350 DATA O2G,O3G
360 DATA 1,1,1,1,2,2,1,1,3,2,1,1
370 '
380 ' Execution
390 '
400 PLAY "O=OCT%(Z);XA$;","XB$;"
410 RETURN
420 '
430 ' Titre
440 '
450 CLS
460 PRINT "ROCK!"
470 PRINT:PRINT"Please, press any key to start"
480 X=RND(1):IF INKEY$="" THEN 480
490 PRINT:PRINT"YEAAH!!"
500 RETURN
```

Commentaires

1. Mesures

Tout Rock'n roll qui se respecte est construit en 12 mesures à 4 temps. Nous avons choisi le Rock en Do (c'est le meilleur). L'enchaînement des accords est le suivant :
Do, Do, Do, Do, Fa, Fa, Do, Do, Sol, Fa, Do, Do
On reconaîtra cette suite dans la ligne 360 où Do est codé 1, Fa est codé 2 et Sol est codé 3.

2. Accords

Les 3 accords sont en DATA dans les lignes 300, 310 et 320. Aux 3 notes de l'accord majeur, nous avons ajouté la septième pour faire couleur locale.

3. Basses

Les notes des basses d'accompagnement sont en DATA dans les lignes 330, 340 et 350. Pour elles, il n'y aura pas de tirage aléatoire. Chaque note est jouée sur les temps forts (1 et 3).

4. Ligne mélodique

Nous avons voulu que les mélodies jouées dans chaque accord soient identiques à une translation près. La construction de la ligne mélodique commune est faite dans la boucle 190-200. Huit notes sont jouées dans chaque accord.
En 200, on choisit une octave au hasard OCT% (I).
En 210, on choisit une hauteur de note HN (I).

5. Exécution

Elle intervient dans les boucles imbriquées de 240 à 280 par l'intermédiaire du sous-programme 380-410.
La boucle I concerne les 12 mesures. La boucle Z les 8 notes de chaque mesure.
N est le numéro de la mesure.
A$ est la note jouée.
B$ est la basse jouée.
La formule en ligne 260 permet de jouer une basse un temps sur deux en suivant les accords. Entre deux notes basses, on joue un silence (R64).

6. Randomize
On reconnaîtra en ligne 480 un simulateur de RANDOMIZE dont le fonctionnement est décrit dans les commentaires du programme précédent (Menuet).

Shebam ! Blop ! Wizzz !

Une note, un bruit, il ne faut pas confondre. Ce n'est pas seulement une affaire de goût mais de physique, c'est-à-dire de forme d'onde. Comme le titre de ce chapitre le laisse deviner, il va s'agir ici de bruit.

Du point de vue programmation, après avoir largement exploité l'instruction PLAY, nous nous attaquons à SOUND (voir Dictionnaire Basic MSX).
Rappelons que SOUND permet de contrôler les 16 registres numérotés de 0 à 15. Les deux registres importants en ce qui concerne le bruit sont les registres 6 et 7. Le registre 6 permet de régler la fréquence du générateur de bruit et le registre 7 le mélange du bruit avec les trois générateurs.

Shebam !

```
10 SOUND 6,50
20 SOUND 7,55
30 SOUND 8,15
```

Cette série de trois instructions provoque un chuintement épouvantable qui évoque aussi bien le départ de la navette spatiale que les chutes du Niagara. Il faut arrêter avec CTRL-STOP.

Commentaires

1. SOUND 6,50
Réglage du niveau du générateur de bruit (assimilable à une fréquence). Le deuxième argument peut prendre une valeur entière 1 et 255. Il faut essayer plusieurs valeurs avant de trouver celle que l'on cherche.

2. SOUND 7,55
La traduction binaire de 55 est 110111. Le seul bit à 0 déclenche le passage du bruit sur le canal A.

3. SOUND 8,15
Règle le son au maximum sur le canal A. Ce canal a été choisi pour l'exemple, on aurait pu aussi bien opter pour le canal B ou C. Notons qu'en faisant sortir le bruit à fond sur les 3 canaux, on n'obtient pas une puissance supérieure.

4. Variations
Elles sont évidemment innombrables avec ce seul bruit. On peut faire varier ensemble le niveau de générateur de bruit (canal 6) et le volume (canal 8 pour A). Mais les variations les plus intéressantes passent par le mélange bruit-son.

Blop !

```
10 SOUND 6,50
20 SOUND 7,55
30 SOUND 8,16
40 SOUND 11,0
50 SOUND 12,100
60 SOUND 13,0
```

Entrez ce programme. Appuyez sur la touche fonction RUN. Normalement ça fait Boum, Blop ou Beng selon les goûts.

Commentaires

1. Bruit
Les lignes 10 et 20 permettent de laisser passer le bruit et le son sur le canal A.
La valeur 16 dans le canal 8 permet de passer sous le contrôle du générateur d'enveloppe (13).

2. Enveloppe
La fréquene est définie sur les canaux 11 et 12 et la forme sur le canal 13. Nous avons fait des choix arbitraires.

3. Variations

Elles sont évidemment presque infinies puisqu'elles peuvent jouer sur tous les paramètres. Nous vous encourageons à tout essayer en notant les configurations intéressantes. On peut commencer par changer la forme de l'enveloppe qui donne des résultats parfois étonnants.

C'est l'enveloppe 0 qui provoque l'effet de déflagration car elle redescend très vite au niveau 0.

L'écho, c'est-à-dire la durée de l'explosion se règle sur le canal 12. Plus on l'augmente et plus le bruit durera jusqu'à donner l'impression d'une apocalypse.

On peut aussi ajouter du son en jouant avec les registres 0 et 1 qui définissent la fréquence du son. Il suffit d'ajouter une ligne 5 du type

```
5 SOUND 0,0 : SOUND 1,1
```

et de laisser passer le son sur le canal 7 par

```
20 SOUND 7,54
```

On peut aussi ne laisser passer que le son par

```
20 SOUND 7,62
```

et obtenir un délicieux bruit de clochette grâce à

```
 5 SOUND 0,100 : SOUND 1,0
40 SOUND 11,0
50 SOUND 12,20
```

On trouvera une déflagration particulièrement effrayante dans le programme "Défense active".

Wizzz !

```
10 SOUND 7,62
20 SOUND 8,15
30 FOR I=255 TO 0 STEP -1
40 SOUND 0,I
50 NEXT I
60 SOUND 8,0
```

Commentaires

10. Le nombre 62 déposé dans le registre 7 a pour effet de laisser passer le son sur le seul canal A. En effet 62 s'écrit 111110 en binaire et le 0 sur le dernier bit libère le son pour le canal A.

Variante : en remplaçant SOUND 7,62 par SOUND 7,54, on laissera passer le bruit mélangé au son sur ce même canal A. En effet 54 s'écrit 110110 et les deux bits à 0 concernent le canal A. Le 4e bit mis à 0 laisse passer le bruit.

20. Le nombre 15 dans le registe 8 met le son au maximum. Inutile d'expliquer pourquoi.

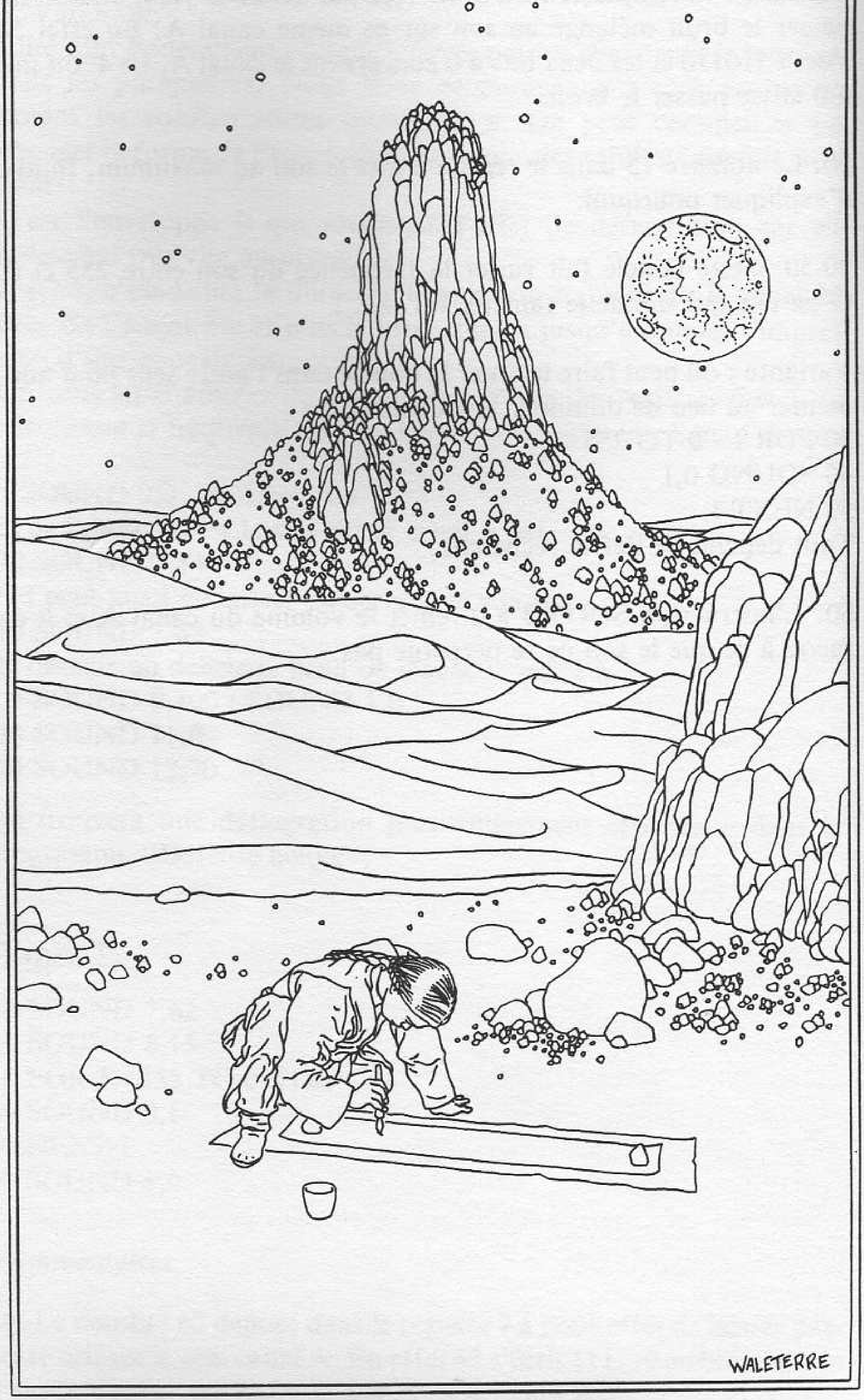
30-50. Cette boucle fait varier la fréquence du son entre 255 et 0. C'est elle qui simule le fameux WIZZZ !

Variante : on peut faire tourner la boucle dans l'autre sens pour augmenter au lieu de diminuer la fréquence.

```
30 FOR I=0 TO 255
40 SOUND 0,I
50 NEXT I
```

Tout dépend de l'effet recherché.

60. L'instruction SOUND 8,0 remet le volume du canal A et 0 de façon à ce que le son ne se perpétue pas.


WALETERRE

Calculs

Un ordinateur est avant tout un calculateur. Les informaticiens le savent. Bien que l'ordinateur domestique n'ait pas pour vocation première de servir au calcul scientifique, il ne faut pas négliger le fait que tout citoyen honnête, ne serait-ce que quand il paye ses impôts, est en mesure d'apprécier les services d'un calculateur rapide et efficace.

Voici donc 5 programmes qui font jongler le micro-ordinateur avec les nombres. Et pour atténuer l'aridité du sujet, nous y avons ajouté un jeu ; de nombres évidemment.

Base besogne

Hé non ! Point de jeu d'aventure, ni d'action quelque peu dégradante.
Mais il faut voir la réalité en face : notre bonne vieille base 10, familière, pratique, nous fait souvent oublier qu'elle est loin d'être la seule et loin de prétendre à l'hégémonie.

En particulier, tout informaticien azerty manipule en base 2 (binaire), 8 (octale) et 16 (hexadécimale), quand il veut parler vite au microprocesseur.
La puce lui répond en binaire, la seule langue qu'elle pratique.
A une question du type ''Savez-vous planter les choux ?'' une puce insolente risque de vous répondre :

01010011	01001110	00100000
01100101	01110010	01110010
01111010	01110010	

Langue de bois des architectures modernes, sachons la débusquer !
Ce programme quant à lui, propose de la manipuler.

```
10 '  **** CONVERSION D'UN NOMBRE
20 '        EN BASE X EN UN NOMBRE
30 '              EN BASE X'          ****
40 '
50 '        ---------------------------
60 '
70 ' Maximum decimal : 32767
80 ' Maximum base 25 : 22AH
90 '
100 '        -------------------------
110 '
120 '
130 '        Initialisation
140 '
150 CLS:KEYOFF:DIM V(15)
160 ON ERROR GOTO 560
170 N$="0123456789ABCDEFGHIJKLMNOPQ"
180 INPUT "Base initiale";B1
190 INPUT "Base de conversion";B2
200 '
210 '        Preparation de l'ecran
220 '
230 CLS:PRINT"Nombres"
```

```
240 PRINT"en base";B1
250 LOCATE 15,0:PRINT"Nombres"
260 LOCATE 15,1:PRINT"en base";B2
270 '
280 '      Boucle principale
290 '
300  FOR I=4 TO 22
310   LOCATE 0,I:INPUT N1$
320   L=LEN(N1$):ND=0
330 '
340 '      Conversion decimale
350 '
360    FOR J=1 TO L
370     V(J)=B1^(L-J)*
       (INSTR(N$,MID$(N1$,J,1))-1)
380    NEXT
390 '
400    FOR J=1 TO L:
        ND=ND+V(J):
       NEXT
410 '
420 '     Conversion base B2
430 '
440  NB2$=""
450  R=(ND MOD B2)+1:ND=INT(ND/B2)
460  NR$=MID$(N$,R,1):NB2$=NR$+NB2$
470  IF ND>=B2 THEN 450 ELSE ND=ND+1
480  NR$=MID$(N$,ND,1):NB2$=NR$+NB2$
490 '
500 '     Affichage de NB2$
510 '
520  LOCATE 15,I:PRINT NB2$
530 '
540  NEXT
550 GOTO 210
560 RESUME 210
```

Commentaires

Plus que les bases, ce programme manipule les chaînes de caractères. Les calculs proprement dits ne portent que sur des décimaux ; c'est là l'artifice, le secret partagé du programmeur et de l'ordinateur.
Tel quel, il ne saura traiter que des nombres entrés dans la fourchette de la base 2 à la base 25.
C'est largement suffisant, pour ne pas dire inutile d'aller au-delà ; cependant, pour les aventuriers, tout reste possible en ne modifiant qu'une ligne de ce programme. Nous vous la laissons deviner...

Passé l'amusement ou la curiosité de contempler l'écriture d'un nombre en base inhabituelle, ce programme peut être utile aux fanatiques

des poke, aux adeptes des tests dans les registres internes, à tous ceux atteints de quelque octetomanie.

Organisation du programme

• N$ en ligne 170 est une variable alphanumérique initialisée par une suite de 26 symboles, nécessaires à l'écriture d'un nombre en base 25.

• Le programme principal tient dans la douzaine de lignes 310-380.
• Le nombre à convertir est systématiquement traduit en base 10 par le passage dans les 2 courtes boucles suivantes :
— la première (360-380) extrait chaque symbole du nombre à convertir et, en fonction du rang qu'il occupe dans N$, lui attribue une valeur décimale par un calcul savant.
— la seconde (400) fait la somme décimale des symboles précédents.
• De 440 à 480, la même méthode est mise en œuvre, cette fois à rebrousse-poils, de façon à obtenir une écriture du nombre en base 10 dans la base de conversion désirée : le reste de la division entière du nombre décimal par la base choisie est calculé autant de fois que nécessaire, et, à chaque fois, transformé par le symbole correspondant dans N$ (ligne 460) ; la nouvelle écriture, pas à pas rangée dans une nouvelle chaîne (NB2$) se fait jusqu'à épuisement.

Remarque

En ligne 160, ON ERROR GOTO 560 déroute le déroulement séquentiel du programme à chaque fois qu'une erreur quelconque se produit, évitant ainsi que le programme "plante".
La ligne 560 ne gère pas à proprement parler l'erreur détectée mais se contente de la masquer par un renvoi en tête.
Cette instruction est placée dans ce programme en prévision d'un nombre incorrect entré au clavier (0>X>32767)

Suggestion

Il est simple et sans doute très utile de modifier le début du programme de façon à obtenir le code ASCII binaire, décimal et hexa de n'importe quelle touche frappée au clavier.

Éditeur

Nous donnons ici une routine qui pourra servir à autre chose qu'à des calculs numériques. Nous l'avons placé ici car nous nous en servirons dans le programme suivant qui permet d'effectuer de grosses multiplications. Mais ses caractéristiques lui autorisent de multiples applications.

Si le programme que vous envisagez d'écrire est interactif, vous utiliserez soit l'instruction INPUT, soit INKEY$.
Si vous choisissez INPUT, on peut conclure deux choses :
— Il ne s'agit pas d'un jeu rapide.
— Vous faites totalement confiance à l'utilisateur.
Avec INPUT, vous ne pourrez contrôler la suite des touches frappées qu'après l'appui de la touche ENTRÉE. L'utilisateur prendra le temps qu'il voudra, mais plus grave, il tapera tous les caractères qu'il voudra et autant qu'il le voudra.
Tout porte donc à s'intéresser plutôt à INKEY$ qui ne présente pas les inconvénients de INPUT que nous venons de souligner. En revanche, deux inconvénients apparaissent lors d'une prise sous INKEY$, l'un mineur, l'autre majeur. L'inconvénient mineur concerne l'absence d'affichage. On y remédie facilement, si besoin est, en faisant suivre les tests d'un ordre PRINT.
En revanche, avec INKEY$, on perd l'avantage de l'éditeur présent sous INPUT. Avec cette dernière instruction, l'utilisateur a en général la possibilité de ramener le curseur en arrière pour corriger certains caractères avant de valider.

Le programme suivant a pour ambition de rassembler les avantages d'INPUT et de INKEY$ sans les inconvénients de l'un et de l'autre.

Fonctionnalité de l'éditeur

• Il filtre les entrées au clavier en n'autorisant la frappe que des touches du clavier correspondant à une tranche du code ASCII.

• Il fixe la position de la fenêtre de saisie (masque) et la longueur maximale de la chaîne saisie.

• Il permet les corrections grâce à un éditeur de ligne simple. Le retour efface le caractère précédent. C'est évidemment moins bien

que l'éditeur Basic. Le lecteur à qui le cœur en dit pourra réécrire un éditeur qui simule l'éditeur Basic (Effacement, Insertion) et qui filtre le clavier, mais nous devons l'avertir : ce n'est pas facile et surtout, écrit en Basic, un tel éditeur risque d'être d'une lenteur insupportable.

```
10 '
20 '  Editeur de ligne
30 '  Saisie sans INPUT
40 '
50 '  Parametres d'entree:
60 '  LMA:nombre maximum de caracters saisis
70 '  LX,LY:Coordonees de debut de saisie
80 '  CINF,CSUP:Bornes des codes permis
90 '
100 ' Parametre de sortie
110 ' REP$:Chaine saisie
120 '
130 ' Exemple:saisie d'un nombre de 10 chiffres
140 '
200 LMA=10:LX=5:LY=10:CINF=48:CSUP=57:REP$=""
210 CLS:PRINT "Donnez un nombre de 10 chiffres"
220 GOSUB 10000
230 PRINT:PRINT "Votre nombre est:"; REP$
999 END
10000 '
10010 '  Editeur de ligne
10020 '
10025 LOCATE LX,LY
10030 R$=INKEY$
10040 IF R$="" THEN 10030
10050 C=ASC(R$)
10060 IF C=13 THEN 10999
10070 IF STICK(0)=7  THEN 10200
10080 IF C<CINF OR C>CSUP THEN 10030
10100 IF LEN(REP$)=LMA THEN 10030
10110 PRINT R$;:REP$=REP$+R$:GOTO 10030
10200 IF POS(1)=LX THEN 10030
10210 PRINT CHR$(29)+CHR$(32)+CHR$(29);
10220 REP$=LEFT$(REP$,LEN(REP$)-1):GOTO 10030
10999 RETURN
```

Commentaires

1. Ce programme est un exemple d'utilisation du sous-programme Éditeur. En tapant RUN, vous pouvez saisir à partir des coordonnées 5, 10, un nombre de 10 chiffres maximum.
Le sous-programme intéressant est entre les lignes 10000 et 10999 et peut être utilisé tel que (par MERGE), dans n'importe quel pro-

gramme. C'est ce que nous faisons dans le programme suivant (Grosses opérations).

2. Paramètres d'entrée
• LMA : un nombre entier fixant le nombre maximum de caractères saisis. LMA ne doit pas dépasser 255.

• LX et LY sont les coordonnées du début de saisie (entiers). LX ne doit pas dépasser 39 et LY ne doit pas dépasser 24.

• CINF et CSUP sont les bornes inférieures et supérieures de la tranche du code ASCII correspondant aux caractères permis.

Exemples :
Nombres entiers : CINF = 48 ; CSUP = 57
Expressions mathématiques : CINF = 40 ; CSUP = 62
Lettres majuscules : CINF = 65 ; CSUP = 90
Tous caractères : CINF = 0 ; CSUP = 255

• REP$ contiendra la chaîne des caractères saisis après concaténation. En appelant le sous-programme, on prendra soin d'écrire REP$ = "".

3. Résultat
Le seul résultat sorti du sous-programme est REP$.

4. Le sous-programme pas à pas
10025 : place le curseur en LX, LY
10030 : lit le caractère frappé au clavier (R$)
10040 : tant que rien n'est appuyé, revenir en 10030.
10050 : lire le code ASCII du caractère.
10060 : si c'est la touche ENTRÉE, sortir.
10070 : si c'est la touche retour du curseur, alors aller en 10200.
10080 : si le caractère n'est pas dans la tranche, retourner en 10030 (saisie d'un caractère).
10100 : si la longueur maximale est atteinte, retouner en 10030 (refus du caractère autre que retour du curseur).
10110 : afficher le caractère saisi ; l'ajouter à la chaîne REP$; retouner en 10030.
10200 : si le curseur est en LX (position initiale), refuser le retour curseur en retournant en 10030.

10210 : revenir en arrière, afficher un blanc (effacer), revenir en arrière.
10220 : enlever dans REP$, le dernier caractère puis retourner en 10030.
10999 : Retourner au programme principal.

Grosses opérations

Certains débutants sont parfois surpris de constater que les micro-ordinateurs ne savent pas faire complètement une multiplication dont le nombre de chiffres au résultat dépasse sept.
Par exemple, si l'on demande le résultat de 12343*778, le nombre qui nous est servi est 9.60285 E+06 qu'il faut comprendre comme 9 602 850, alors que n'importe quel bambin de l'école primaire sait que l'exacte réponse est 9 602 854. Il est vrai que quatre unités devant plus de 9 000 000, ça ne pèse pas lourd !
Le programme suivant donne le produit de deux nombres entiers dont le nombre de chiffres peut aller jusqu'à 50 et plus ! Il utilise la routine de saisie des chiffres construite dans le chapitre précédent et exécute les calculs dans un tableau, exactement comme on le fait à la main. C'est un peu long de l'exécuter, mais très spectaculaire.

```
10 '
20 '   Grosses operations
30 '
90 LMA=35:DIM F(2,LMA),P(2*LMA)
100 CLS
110 PRINT"Donnez les facteurs de la"
115 PRINT "multiplication (MAX:35 chiffres"
120 PRINT:PRINT"Premier facteur:"
130 LMA=35:LX=0:LY=4:CINF=48:CSUP=57:REP$=""
140 GOSUB 10000:L1=LEN(REP$)
145 LL=L1:N=1:GOSUB 1000
150 PRINT:PRINT:PRINT"Deuxieme facteur:"
160 LY=7:REP$=""
170 GOSUB 10000:L2=LEN(REP$)
175 LL=L2:N=2:GOSUB 1000
200 '
210 '  Calcul
220 '
230 FOR I=L1 TO 0 STEP -1
240 R=0:K=L1-I+1
```

```
250 FOR J=L2 TO 0 STEP -1
260 P=F(1,I)*F(2,J)+R
270 R=INT(P/10)
280 P(K)=P(K)+P MOD 10
290 P(K+1)=P(K+1)+INT(P(K)/10)
300 P(K)=P(K) MOD 10:K=K+1
310 NEXT J:NEXT I
400 '
410 '  Affichage resultat
420 '
430 PRINT:PRINT:PRINT"Resultat:"
440 FOR I=L1+L2 TO 1 STEP -1
450 IF I=L1+L2 AND P(I)=0 THEN 470
460 PRINT CHR$(48+(P(I)));
470 NEXT I
999 END
1000 '
1010 ' Construction tableau
1020 '
1030 FOR I=1 TO LL
1040 F(N,I)=VAL(MID$(REP$,I,1))
1050 NEXT I
1999 RETURN
10000 '
10010 '  Editeur de ligne
10020 '
10025 LOCATE LX,LY
10030 R$=INKEY$
10040 IF R$="" THEN 10030
10050 C=ASC(R$)
10060 IF C=13 THEN 10999
10070 IF STICK(0)=7  THEN 10200
10080 IF C<CINF OR C>CSUP THEN 10030
10100 IF LEN(REP$)=LMA THEN 10030
10110 PRINT R$;:REP$=REP$+R$:GOTO 10030
10200 IF POS(1)=LX THEN 10030
10210 PRINT CHR$(29)+CHR$(32)+CHR$(29);
10220 REP$=LEFT$(REP$,LEN(REP$)-1):GOTO 10030
10999 RETURN
```

Commentaires

1. Appels de l'éditeur

Le sous-programme d'édition décrit dans le chapitre précédent est à partir de la ligne 10000.

Le programme principal l'appelle 2 fois. La première fois en ligne 130. La deuxième fois en ligne 160.

2. Construction des tableaux

Les nombres destinés à être les deux facteurs de la multiplication sont rendus par le sous-programme 10000 sous la forme d'une chaîne

REP$. Il s'agit ensuite de transformer cette chaîne en un tableau de chiffres F. C'est la fonction du sous-programme 1000 qui extrait chaque caractère de REP$ et en met la valeur (VAL) dans le tableau F.
F(1,I) contient le Ie chiffre du 1er facteur.
F(2,I) contient le Ie chiffre du 2e facteur.

3. Calcul
La partie calcul est entre les lignes 200 et 400. Si vous entrez des nombres de plus de 10 chiffres, le temps de calcul peut être important. Il faut être patient. Les calculs reproduisent la méthode "à la main" à ceci près que l'addition des produits partiels est faite au fur et à mesure et non à la fin.
P contient les produits chiffre à chiffre.
R est la retenue.
P(K) contient la somme des produits dans la colonne K.
K est le décalage à gauche.

4. Affichage du résultat
L1 + L2 est le nombre maximal de chiffres du produit. Si le premier chiffre est 0, on ne l'écrit pas (ligne 450).
Nous ne nous sommes pas contentés de l'instruction PRINT P(I) car le format d'impression des entiers en Basic MSX prévoit un blanc devant et un derrière, ce qui n'est pas très esthétique lorsque plusieurs chiffres doivent être affichés à la suite.
La ligne 460, par l'intermédiaire du code ASCII et puisque nous sommes sûrs que P(I) est un entier inférieur à 10, affiche le bon chiffre sans espace (48 est le code de 0).

Intérêt et principal

Vous voulez acheter un micro-ordinateur, remplacer votre voiture, construire une maison... Il vous faut un peu, beaucoup d'argent.
Mais, cigale, vous n'avez aucune économie, et vous êtes contraint d'emprunter à une fourmi : votre banquier, ou votre tonton à héritage. Qui dit emprunt dit taux d'intérêts, remboursements, amortissement... Vous n'y connaissez rien, mais ne voulez pas vous engager à l'aveuglette : ayez une idée plus claire de vos engagements en utilisant notre programme.

De toute façon, il vous faudra rembourser, inérêt et principal, foi d'animal.

Nous nous intéressons, dans ce programme, au cas d'un emprunt à **mensualités constantes**, ce qui représente une situation courante.

Quatre nombres interviennent :
- **La somme empruntée**, le « Capital », que nous noterons *C*.
- **Le taux d'intérêt**, annuel, ici nommé *I*.
- **Le montant de chaque mensualité**, noté *M*.
- **Le nombre de mensualités**, noté *N* : si vous empruntez sur 7 ans, le nombre N vaut 7 × 12, c'est-à-dire 84.

Selon les situations concrètes, quatre problèmes peuvent vous tracasser :

Problème 1 : N ?
J'ai besoin immédiatement d'une certaine somme *C*, le taux d'intérêt *I* m'est imposé, et je peux rembourser *M* francs par mois :

Pendant combien de mois devrais-je rembourser ?

Problème 2 : M ?
J'ai besoin de la somme *C*, le taux d'intérêt est *I* donné, et je voudrais avoir fini de rembourser en *N* mois.

Combien devrais-je payer par mois ?

Problème 3 : C ?
Je suis prêt à payer *M* francs par mois, pendant *N* mois. Le taux *I* est donné.

Combien puis-je emprunter ?

Problème 4 : I ?
J'ai besoin d'une certaine somme *C*, et l'on me propose de verser *M* francs par mois, pendant *N* mois.

Quel taux d'intérêt me fait-on payer ?

• Après chaque calcul est offert, en prime, la **table d'amortissement** de votre emprunt : pour chaque échéance est déterminée la part de la mensualité *M* qui relève du remboursement du capital (le principal) et la part « perdue », qui relève des intérêts ; vous verrez aussi que cette dernière diminue de plus en plus ; vous empruntez ? N'en soyez pas tout amorti.

```
10 '
20 ' Interet et principal
30 '
40 SCREEN 0:KEY OFF:ON ERROR GOTO 15000
50 PRINT "INTERET ET PRINCIPAL"
60 PRINT:PRINT "Vous desirez calculer:":PRINT
70 PRINT"1: Le nombre de mensualites"
80 PRINT"2: Le montant d'une mensualite"
90 PRINT"3: Le montant de l'emprunt"
100 PRINT"4: Le taux d'interet"
110 A$=INPUT$(1):A=VAL(A$)
120 IF A<1 OR A>4 THEN 110 ELSE CLS
130 ON A GOSUB 1000,2000,3000,4000
140 PRINT:PRINT "Appuyez sur une touche."
150 A$=INPUT$(1)
160 GOSUB 5000
999 END
1000 '
1010 '  Nombre de mensualites
1020 '
1030 PRINT "Nombre de mensualites"
1040 PRINT:INPUT "Taux d'interet annuel (%):";I
1045 IF I>99.99 OR I<=0 THEN 1040
1050 INPUT "Montant de l'emprunt:";C
1060 INPUT "Montant d'une mensualite:";M
1070 I=I/100:IM=(I+1)^(1/12)-1
1080 N=-LOG(1-IM*C/M)/LOG(1+IM)
1090 N=INT(N)
1100 PRINT:PRINT "Vous devrez rembourser en";N;"mois."
1999 RETURN
2000 '
2010 ' Montant d'une mensualite
2020 '
2030 PRINT "Montant d'une mensualite"
2040 PRINT:INPUT "Taux d'interet annuel (%):";I
2045 IF I>99.99 OR I<=0 THEN 1040
2050 INPUT "Montant de l'emprunt:";C
2060 INPUT "Nombre de mensualites:";N
2065 IF N>300 THEN 2060
2070 I=I/100:IM=(1+I)^(1/12)-1
2080 M=C*IM/(1-(1+IM)^(-N))
2090 PRINT:PRINT "Vous devrez rembourser"
2100 PRINT USING "######.##";M;:PRINT" Francs par mois."
2999 RETURN
3000 '
```

```
3010 ' Montant de l'emprunt
3020 '
3030 PRINT "Montant de l'emprunt"
3040 PRINT:INPUT "Taux d'interet annuel (%):";I
3045 IF I>99.99 OR I<=0 THEN 1040
3050 INPUT "Montant d'une mensualite:";M
3060 INPUT "Nombre de mensualites:";N
3065 IF N>300 THEN 2050
3070 I=I/100:IM=(1+I)^(1/12)-1
3080 C=M*(1-(1+IM)^(-N))/IM
3090 PRINT:PRINT "Vous pouvez emprunter"
3100 PRINT USING "#######.##";C;:PRINT" Francs."
3999 RETURN
4000 '
4010 ' Taux d'interet annuel (%)
4020 '
4030 PRINT "Taux d'interet annuel (%)"
4040 INPUT "Montant de l'emprunt:";C
4050 INPUT "Montant d'une mensualite:";M
4060 INPUT "Nombre de mensualites:";N
4070 IF N>300 THEN 4060
4080 K=C/M:I=1/K-K/(N*N)
4090 Z=(1+I)^(-N):F=(1-Z)/I-K
4100 FF=N*Z/((1+I)*I)+(Z-1)/(I*I)
4110 I=I-F/FF
4120 IF ABS(F)>1E-04 THEN 4090
4130 IM=I:I=(1+I)^12-1:I=100*I
4140 PRINT:PRINT"Le taux d'interet est";
4150 PRINT USING " ##.##";I;:PRINT "%"
4999 RETURN
5000 '
5010 ' Table d'amortissement
5020 '
5030 CLS:C1=C
5040 PRINT"  ECHEANCE   INTERET   CAPITAL"
5050 PRINT STRING$(37,"-")
5060 FOR K=1 TO N
5070 IT=C1*IM:PPAL=M-IT
5080 PRINT SPC(7);:PRINT USING "###";K;
5090 PRINT USING "#######.##";IT;PPALN
5110 C1=C1-PPAL
5120 IF CSRLIN<20 THEN 5200
5130 PRINT:PRINT "Appuyez sur une touche."
5140 A$=INPUT$(1):CLS
5150 PRINT"  ECHEANCE   INTERET   CAPITAL"
5160 PRINT STRING$(37,"-")
5200 NEXT K
5999 RETURN
15000 '
15010 ' Calcul impossible
15020 CLS
15030 PRINT "Le calcul est impossible."
15040 PRINT "Changez les donnees."
15100 END
```

Commentaires

1. Calculs

• Les quatre variables utilisées dans le programme sont donc C, M, N et I ; remarquons d'abord qu'il faut passer de l'intérêt annuel I à son équivalence mensuelle IM : c'est ce qui est fait par exemple à la ligne 1050.

Indiquons maintenant comment on peut retrouver la "formule" qui lie ces quatre variables. Pour cela, appelons C_K la fraction de capital restant à rembourser après la k-ième échéance (appelée parfois valeur résiduelle). On a bien sûr $C_O = C$ et $C_N = 0$. Ces nombres sont d'ailleurs les valeurs prises par la variable C1 du sous-programme 5000.

Examinons maintenant le lien entre C_K et C_{K+1}. Ce calcul est fait dans le sous-programme 5000, à chaque passage dans la boucle : la somme versée mensuellement M sert à payer les intérêts sur la valeur résiduelle C_K et à rembourser une partie du capital. On a donc :

$$C_{K+1} = C_K - (M - IM*C_K)$$

cette égalité peut aussi s'écrire :

$$\boxed{C_{K+1} - \frac{M}{IM} = (1 + IM)\,(C_K - \frac{M}{IM})}$$

on en déduit donc aisément :

$$C_N - \frac{M}{IM} = (1 + IM)^N\,(C - \frac{M}{IM})$$

Il ne reste plus qu'à poser $C_N = 0$ pour obtenir la formule :

$$\boxed{M = \frac{C*IM}{1 - (1 + IM)\uparrow(-N)}}$$

• Remarquons que dans cette formule, seule la variable IM apparaît deux fois : il n'est pas possible de l'exprimer directement en fonction des trois autres ; c'est pour cela qu'il est nécessaire d'employer une méthode de résolution d'équation par approximation : c'est ce qui est fait dans le sous-programme 4000..Sans entrer dans les détails, signalons que nous avons appliqué la méthode de Newton à la fonction notée F ; la fonction dérivée est notée FF.

• Remarquons également que dans le sous-programme 1000, le nombre d'échéances N est arrondi à l'entier immédiatement inférieur ; attendez-vous donc à payer une dernière mensualité un peu plus forte que les autres.

• Signalons enfin qu'il est parfois utilisé le taux annuel IA défini par IA = 12*IM. Cela est notamment le cas quand il avantage la banque : à vous d'être attentif...

2. Erreurs

Ligne 40, on reconnaît l'instruction ON ERROR qui branche sur la ligne 15000 lorsqu'une erreur est détectée.
Il s'agira ici nécessairement d'une erreur de calcul. Elle peut être la conséquence d'incompatibilité entre les paramètres fournis par l'utilisateur. Impossible en effet de rembourser un emprunt de plusieurs milliards en payant des mensualités de 10 F !
D'autre part, le nombre de mensualités est limité à 300. Aucun banquier ne prête sur plusieurs siècles !
Le programme ne gère pas l'origine des erreurs. Un aménagement dans ce sens est possible. Pour le réaliser, consultez les articles ON ERROR GOTO et RESUME dans le dictionnaire Basic MSX.

3. Changement de page

Le sous-programme 5000 affiche la table d'amortissement sur 20 lignes. Le test en ligne 5120 sert à continuer dans la boucle tant que la 20ᵉ ligne n'est pas atteinte : CSRLIN donne le numéro de ligne du curseur d'écriture. Lorsque la limite est atteinte, on demande à l'utilisateur de frapper une touche pour obtenir la suite du tableau.

4. Ligne

L'instruction en ligne 5050 affiche une ligne de tirets horizontaux. Le premier argument de STRING$ donne le nombre de répétitions du caractère donné en second argument.

Biorythme

Faut-il y croire ou pas ?
L'honnêteté nous oblige à avouer que nous ne sommes pas des inconditionnels du biorythme. Pourtant, le jour où nous avons écrit ce programme, notre sinusoïde intellectuelle était à son maximum.
Le programme devrait donc être particulièrement performant.
Ce qui nous rappelle une remarque célèbre : "Je ne suis pas superstitieux, ça porte malheur !"

La vogue des biorythmes nous vient des U.S.A. et du Japon où ils sont, paraît-il, pris aux sérieux. Il s'agit d'une théorie selon laquelle l'activité humaine de chaque individu obéit à des lois cycliques, les hauts et les bas se reproduisant périodiquement.

Les trois principaux secteurs de l'activité : physique, émotionnel, intellectuel ont pour période respective 23, 28 et 33 jours. Au jour de sa naissance, chaque personne est au niveau moyen pour les trois secteurs. Puisqu'il s'agit de périodes exactes, il suffit, pour une personne donnée, de connaître le nombre de jours écoulés depuis sa naissance pour en déduire l'état de son biorythme.

Nous proposons un programme qui visualise l'état biorythmique d'un individu pour un mois entier, en représentant l'alternance de hauts et de bas, par des sinusoïdes, de couleurs différentes pour chaque secteur d'activité.

On pourra relever dans le listing quelques routines intéressantes :
— saisie de données
— calcul du nombre de jours entre deux dates
— tracé de sinusoïde
— écriture dans l'écran graphique

```
10 '
20 '   Biorythme
30 '
40 SCREEN 0:COLOR 10,1,1
50 GOSUB 30000
60 N=N-N1:IF N<0 OR N>32776! THEN PRINT
   "Impossible":GOTO 999
```

```
70 PH=N MOD 23
80 EM=N MOD 28
90 IN=N MOD 23
100 SCREEN2:OPEN"GRP:"FOR OUTPUT AS #1
110 GOSUB 1000
120 GOSUB 2000
998 GOTO  998
999 END
1000 '
1010 ' Decor
1020 '
1030 LINE (47,5)-(256,170),10,B
1040 LINE (47,85)-(256,85),10
1050 FOR X=47 TO 256 STEP 7
1060 LINE (X,85)-(X,82),10
1070 NEXT X
1080 PSET (50,176),1:COLOR 6
1090 PRINT #1,"Physique ";
1100 COLOR 4:PRINT #1," Emotionnel"
1110 PSET (50,185),1
1120 COLOR 15:PRINT #1,"Intellectuel";
1999 RETURN
2000 '
2010 '  Trace des courbes
2020 '
2030 PI=3.14159:K=2*PI/23
2040 FOR X=0 TO 206
2050 PSET (X+48,85-75*SIN(K*(X/7+PH))),6
2060 NEXT X
2070 K=2*PI/28
2080 FOR X=0 TO 206
2090 PSET (X+48,85-75*SIN(K*(X/7+EM))),4
2100 NEXT X
2110 K=2*PI/33
2120 FOR X=0 TO 206
2130 PSET (X+48,85-75*SIN(K*(X/7+IN))),15
2140 NEXT X
2999 RETURN
10000 '
10010 '  Ecart dates
10020 '
10030 BIS=0:IF A MOD 4=0 THEN BIS=1
10040 CR=0:IF M=1 OR M=2 THEN CR=2-BIS
10050 N=INT(365.25*A)+INT(30.56*M)+J+CR
10999 RETURN
30000 '
30010 '  Saisie des donnees
30020 '
30030 PRINT "Annee de naisance:";
30040 INPUT A:A=INT(A)
30050 IF A<0 OR A>2500 THEN 30030
30060 PRINT "Mois de naissance (1 a 12):";
30070 INPUT M:M=INT(M)
30080 IF M<1 OR M>12 THEN 30060
30090 PRINT "Jour de naissance (1 a 31):";
```

```
30100 INPUT J:J=INT(J)
30110 IF J<1 OR J>31 THEN 30090
30120 GOSUB 10000:N1=N
30130 PRINT:PRINT "Annee de biorythme:";
30140 INPUT A:A=INT(A)
30150 IF A<0 OR A>2500 THEN 30130
30160 PRINT "Mois de biorythme (1 a 12):";
30170 INPUT M:M=INT(M)
30180 IF M<1 OR M>12 THEN 30160
30190 J=1:GOSUB 10000
30999 RETURN
```

Commentaires

1. Saisie des données

Le sous-programme 30000 saisit la date de naissance et la date de biorythme. Les données sont contrôlées par un test qui vérifie qu'elles se situent dans le bon encadrement.

Ce test est une garantie suffisante pour s'assurer que les valeurs transmises au programme de calcul ne le planteront pas.

2. Intervalle entre deux dates

C'est l'objet du sous-programmes 10000.

BIS est une variable qui repère les années bissextiles. CR est une variable qui corrige le nombre de jours du mois.

N est le nombre de jours séparant les deux dates. On reconnaîtra au passage le nombre moyen de jours par année (365.25) et le nombre moyen de jours par mois (30.56).

Cette formule n'est pas d'une exactitude rigoureuse.

3. Tracé de sinusoïde

Les trois sinusoïdes sont tracées dans le sous-programme 2000. On affiche un point dans chaque colonne de la fenêtre graphique.

Les couleurs (6, 4, 15) permettent de s'y retrouver.

On observera lors du déroulement que les couleurs "bavent" les unes sur les autres. Il ne s'agit pas d'un défaut du programme mais d'un défaut de la machine. Très exactement, il s'agit d'un défaut de mémoire, au sens manque de mémoire. Chaque groupe de 8 pixels horizontaux n'accepte que deux couleurs, une pour la forme, une pour le fond. Tout pixel de ce groupe allumé dans une troisième couleur contamine les autres. On appelle ce phénomène la rigidité des

couleurs. Cette contrainte répond à des impératifs techniques qui interdisent de consacrer trop de place à la mémoire d'écran. Dans le cas de notre biorythme, il n'y a rien d'autre à faire que de le regretter.

4. Écriture dans l'écran graphique

A priori, la seule instruction PRINT ne permet pas d'écrire dans l'écran graphique défini par SCREEN2 (ligne 100).
Il est nécessaire d'ouvrir un fichier dans un périphérique dont le code est GRP (ligne 100). Le numéro de canal associé est 1.
L'écriture suit la syntaxe des fichiers ; il suffit de rappeler le numéro du canal concerné (lignes 1090, 1100, 1120).
Un autre problème se présente aussitôt. L'instruction LOCATE, qui place le curseur d'écriture, est sans objet ici puisqu'il n'y a pas de curseur d'écriture. Il faut utiliser le curseur graphique que l'on place grâce à une instruction PSET.

Ce système, bien qu'un peu fastidieux, présente tout de même certains avantages. Il permet d'écrire n'importe où sur l'écran indépendamment des lignes et colonnes habituelles. Il permet également de mêler plusieurs couleurs d'écriture ce qui n'est pas possible dans les écrans d'écriture (0 ou 1).
Dans notre cas, les couleurs servent de légende pour faire écho aux couleurs des sinusoïdes.

Sphynx

Deviner un nombre pensé secrètement par l'adversaire voilà un jeu très connu.
Mais lorsque les informations que vous recevez sont le fruit de calculs complexes, on peut dire avec Napoléon ''ça se corse !''.

Ce jeu, au demeurant plus simple qu'il n'y paraît, change agréablement de l'éternel ''Mastermind''.
Mais attention, ici aussi, l'analyse des essais précédents se révèle fondamentale.

```
10 '        **************
20 '        *            *
30 '        *   SPHYNX   *
40 '        *            *
50 '        **************
60 '
70 '               Entete
80 '
90 CLS:C=1:FOR I=0TO20
100 LOCATEI,I:PRINT"SPHYNX":NEXT
110 LOCATE7,22
120 PRINT"Appuyez sur une touche"
130 '
140 GOSUB 770
150 '
160 GOSUB 850
170 '
180 '
190 SCREEN2
200 OPEN"GRP:" FOR OUTPUT AS #1
210 PRESET(20,20):COLOR C
220 PRINT#1,"QUEL NOMBRE PROPOSEZ-VOUS "
230 '
240 '       Boucle de saisie
250 '       du nombre joueur
260 '
270 N1$=""
280 FOR I=1 TO 4
290 COLOR10:GOSUB 770
300 IF I=1 THEN LINE(T*6,123)-(T*6+20,130),4,BF:
    LINE(140,55)-(100,45),4,BF
310 '
320 '  La touche est-elle un chiffre
330 '
340 IF ASC(R$)<48 OR ASC(R$)>57
    THEN 290 ELSE PRINT#1,R$;
```

```
350 N1$=N1$+R$
360 NEXT
370 '
380 '
390 N1=VAL(N1$)
400 '
410 '    Comparaison des nombres
420 '
430 IF N1<N THEN ECC=N-N1:GOTO 500
440 ECC=N1-N
450 IF ECC=0 THEN GOTO 650
460 '
470 '   Cumul des chiffres de la
480 '          difference
490 '
500 T=0:ECC$=STR$(ECC)
510 FOR I=1 TO LEN(ECC$)
520 T=T+VAL(MID$(ECC$,I,1))
530 NEXT
540 '
550 GOSUB 920
560 '
570 '    Affichage de l'indication
580 '             de distance
590 '
600 COLOR9:PRINT#1,T
610 GOTO 210
620 '
630 '          Bravo !
640 '
650 PRESET(97,55):COLOR0
660 PRINT#1,"GAGNE"
670 '
680 GOSUB 770
690 '
700 '           Encore ?
710 '
720 IF R$="F" OR R$="f" THEN END
730 CLOSE:GOTO160
740 '
750 '      Test du clavier
760 '
770 R$=""
780 R$=INKEY$:X=RND(1)
790 IF R$="" THEN 780
800 RETURN
810 '
820 '      Choix au hasard
830 '     entre 999 et 9999
840 '
850 N=999+INT(RND(1)*9000)
860 PRINTN
870 RETURN
880 '
890 '    Representation graphique
```

```
900 '          de la distance
910 '
920 C=C+1:IF C>15 THEN C=0
930 IF C=4 THEN C=6
940 LINE(20,100)-(T*6,120),C,BF
950 PRESET(T*6,123)
960 RETURN
```

Commentaires

L'ordinateur va tirer au hasard un nombre de 4 chiffres compris entre 999 et 10000.
Vous devez le trouver rien de plus.

Pour cela, la machine vous orientera à chacune de vos propositions en vous fournissant la somme des chiffres de la différence entre votre proposition et le nombre caché.

Organisation du programme

- Les lignes 90-120 contiennent l'en-tête du jeu. Le sous-programme appelé en 140 initialise RND.

- Le déroulement est ensuite orienté vers le tirage au hasard du nombre caché (850-870).

- De 280 à 360 la boucle de saisie de la réponse, sous forme alphanumérique appelle 4 fois le sous-programme de test du clavier.

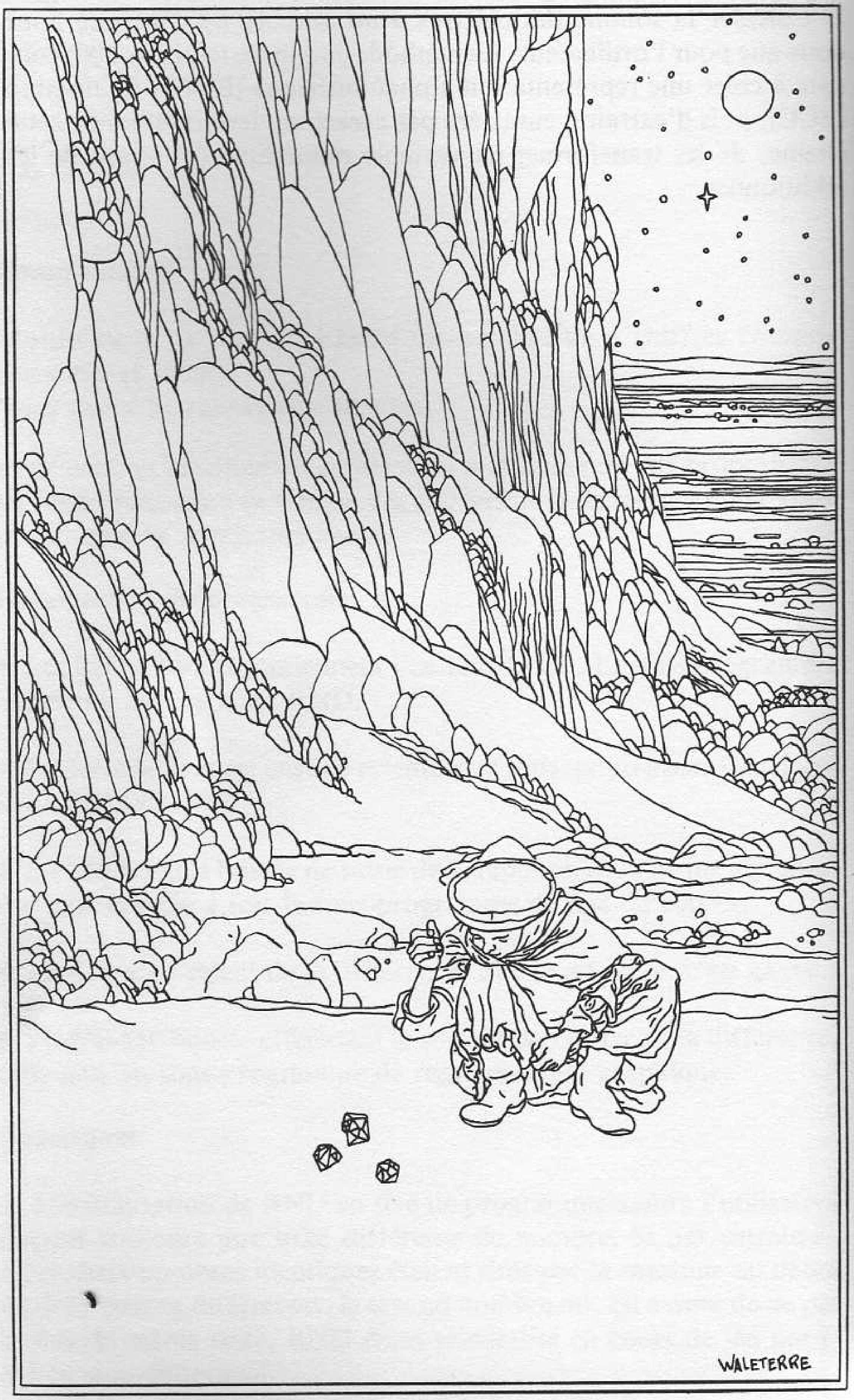
- En 430-450, calcul de la différence. Si elle est nulle, c'est gagné.

- En 500-550 boucle effectuant la somme du chiffre de la différence, puis saut au sous-programme de représentation graphique.

Remarques

1. L'initialisation de RND en tête de programme assure l'utilisateur d'avoir toujours une suite différente de nombre. Si par extraordinaire deux nombres identiques étaient tirés par la machine au début de deux parties différentes, le second nombre lui, est assuré de ne pas être de la même suite, RND étant réinitialisé en cours de jeu par le même sous-programme.

2. Calculer la somme des chiffres d'un nombre est plus aisé pour nous que pour l'ordinateur. La méthode employée ici (ligne 500) consiste à créer une représentation alphanumérique (ECC$) du nombre (ECC), puis d'extraire, caractère par caractère, les symboles de cette chaîne, de les transformer en variable numérique (T) avant de les additionner.


WALETERRE

Jeux

Serpent
Défense active
Miroir aux pixels

Les micro-ordinateurs au standard MSX attirent très probablement les joueurs en tous genres. Les cartouches vendues dans le commerce proposent des jeux d'action très spectaculaires écrits en langage Assembleur. Le Basic ne peut rivaliser avec eux quant à la rapidité d'éxécution. Les programmes que nous fournissons ici ne sont qu'un prétexte à utiliser certaines ressources du Basic MSX. A commencer par l'interactivité (Serpent) grâce à laquelle on peut considérer les quatre touches de déplacement du curseur comme un joystick très satisfaisant. Enfin et surtout, les sprites qui sont les meilleurs outils pour l'animation (Défense active).

Serpent

Ce serpent-là pourrait bien être de la race des ténias puisqu'il s'allonge en mangeant. Il s'agit d'un serpent amateur d'arithmétique car il se nourrit de chiffres et son corps s'allonge d'autant plus que le nombre qu'il avale est important. Les qualités exigées pour pratiquer ce jeu ne sont pas celles d'un calculateur prodige mais tout simplement de bons réflexes, puisqu'il va falloir guider ce curieux animal sur l'écran sans qu'il heurte les bords et sans que sa tête ne vienne buter sur son corps.

A chaque fois que le serpent passe sur un nombre, son corps s'allonge, un autre nombre apparaît et le joueur dispose d'un temps limité avant d'emmener l'animal s'en repaître.

Lorsque le mot serpent apparaît sur l'écran, il faut appuyer sur une touche pour lancer le jeu. Le corps du serpent est au début de longueur 5 et pendant tout le jeu, il sera affiché en haut et à droite de l'écran. Pour atteindre un nombre, il faut faire au maximum 200 reptations. Le compteur des reptations encore disponibles est inscrit en bas et au milieu de l'écran. Tant que vous n'appuyez pas sur une touche pour lancer le jeu, une musique obsédante joue sans cesse. Ensuite, c'est le grand silence de la savane.

Pour déplacer la bête, il suffit d'utiliser les quatre touches de déplacement du curseur à droite du clavier. Il est vivement recommandé de bien les situer au bout des doigts car le joueur devra avoir les yeux rivés sur l'écran et toute fausse manœuvre risque de lui être fatale.

La difficulté apparaît réellement lorsque le corps du serpent atteint une longueur voisine de la centaine et que le nombre à manger vient se loger à l'intérieur des méandres dessinés par les nombreux anneaux. Mieux vaut alors ne pas s'affoler.

```
10 '_______________
20 '_             _
30 '_   serpent   _
40 '_             _
50 '_______________
60 DEFINT A-Z:SCREEN 0:KEY OFF
70 COLOR3,1,4:CLS:LOCATE 10,10:PRINT "SERPENT"
```

```
80 IF INKEY$<>"" THEN 80
90 Z=RND(1):IF INKEY$="" THEN 90
100 '  Initialisation des variables
110 DIM T(40,24)
120 XQ=3:YQ=11:X=8:Y=11:SC=5:F=0:CH$="■":TE$="◆"
130 CLS:FOR X=3 TO 7
140 T(X,11)=2:LOCATE X,11:PRINT CH$;
150 NEXT X: PRINT TE$
160 GOSUB 470
170 GOSUB 520:IF INKEY$="" THEN 170
180 '  Analyse du clavier
190 IF STICK(0)=1 THEN XX=0:YY=-1:C=3
200 IF STICK(0)=3 THEN XX=1:YY=0:C=2
210 IF STICK(0)=5 THEN XX=0:YY=1:C=4
220 IF STICK(0)=7 THEN XX=-1:YY=0:C=1
230 IF X+XX<1 OR XX+X>40 THEN 340
240 IF Y+YY<1 OR YY+Y>22 THEN 340
250 IF T(X+XX,Y+YY)>0 THEN 420
260 T(X,Y)=C:LOCATE X,Y:PRINT CH$
270 X=XX+X:Y=Y+YY:DEP=DEP+1
280 LOCATE X,,Y:PRINT TE$
290 PAS=PAS+1:LOCATE 0,0:PRINT 200-PAS;
300 IF PAS=200 THEN GOSUB 530:GOTO 420
310 IF T(X,Y)>=0 THEN 330
320 U=0:UM=V:SC=SC+V:LOCATE 34,0:PRINTUSING"###";
    SC:PAS=0:GOSUB 480
330 IF U<UM THEN U=U+1:GOTO 180
340 LOCATE XQ,YQ:PRINT " ":CQ=T(XQ,YQ):T(XQ,YQ)=0
350 ' Direction du serpent
360 IF CQ=0 THEN 420
370 IF CQ=1 THEN XQ=XQ-1:YQ=YQ
380 IF CQ=2 THEN XQ=XQ+1:YQ=YQ
390 IF CQ=3 THEN XQ=XQ:YQ=YQ-1
400 IF CQ=4 THEN XQ=XQ:YQ=YQ+1
410 GOTO 180
420 ' Fin de partie
430 FOR I=15 TO 0 STEP -1:COLOR ,I:BEEP:NEXT I
440 GOSUB 520
450 CLS:LOCATE 10,10: PRINT "SCORE:";SC
460 KEY ON:LOCATE 0,0:END
470 ' Determination et affichage du nombre
480 XC=1+RND(1)*36:YC=RND(1)*20+2
490 IF T(XC,YC)<>0 THEN 480
500 V=RND(1)*8+49:LOCATE XC,,YC:PRINT CHR$(V):T(XC,YC)
    =-1:V=V-48
510 RETURN
520 ' Musique
530 L%=64:T%=255
540 PLAY "L=L%;T=T%;O1ABCDEO2ABCDEO4ABCDEO5ABCDE"
550 RETURN
```

Commentaires

1. Écran

La ligne 60 définit toutes les variables en type entier, sélectionne l'écran alphanumérique 40 colonnes et supprime les références aux touches fonction. On observera, en fin de programme (ligne 460) que ces références sont réaffichées par KEY ON pour que le programme puisse être facilement relancé.

2. Déplacements

Le tableau T contient des renseignements sur tous les caractères de l'écran. Si rien n'est affiché aux coordonnées (X,Y), alors T (X,Y) = 0. Si un nombre à manger est affiché, alors T (X,Y) = −1. Si un anneau du corps du serpent est affiché, alors T (X,Y) contient un numéro entre 1, 2, 3 et 4 qui indique la direction de l'anneau précédent : 1 pour la gauche, 2 pour la droite, 3 pour le haut et 4 pour le bas. Ce numéro est un pointeur (C).

A chaque fois qu'une touche du clavier est appuyée, il faut déplacer la tête (coordonnées X et Y), c'est-à-dire imprimer une nouvelle tête et un nouvel anneau à la place de l'ancienne tête. T (X,Y) est alors chargé du pointeur C (ligne 260). Mais il faut également effacer le dernier anneau (la queue du serpent !) dont les coordonnées sont XQ et YQ, pour simuler le déplacement de tout le corps.

Pour connaître les nouvelles coordonnées du dernier anneau, il suffit de lire le pointeur et de recalculer XQ et YQ (lignes 370 à 400). A chaque tour, seules la tête et la queue sont prises en compte sans qu'il soit nécessaire de dessiner l'ensemble des anneaux.

Défense active

Voici les sprites (lutins). Il s'agit d'objets graphiques faciles à animer. On les utilise essentiellement pour les jeux d'action. Nous ne faillirons pas à cette tradition en proposant un jeu du type Space Invaders.

L'important pour la programmation de ce genre de jeu est de bien comprendre les fonctions et les modes d'utilisation des sprites et d'enrichir proressivement l'action et les règles. Pour celà, il suffit de faire preuve d'imagination et d'organisation dans l'écriture du programme.

Nous vous conseillons d'abord de consulter les articles faisant référence à la gestion des sprites dans le dictionnaire Basic MSX.

- SPRITE ON/OFF/STOP
- SPRITE $
- ON SPRITE GOSUB

Le programme qui suit vous met aux commandes d'un lanceur de missiles capable de détruire des vaisseaux ennemis patrouillant en tous sens en haut de l'écran.
Vous devrez en détruire le plus possible et faire mouche le plus souvent possible. Votre survie en dépend.

```
10 '
20 ' Jeu d'action
30 '
40 SOUND 7,28:SOUND 6,25:SOUND 10,16
50 SOUND 11,0:SOUND 12,50
60 GOSUB 480
70 S=1:RESTORE 700:GOSUB 580
80 S=2:RESTORE 750:GOSUB 580
90 S=3:RESTORE 800:GOSUB 580
100 SC=20:X1=112:Y1=168:X3=X1:Y3=Y1-1
110 X2=0:Y2=1:X4=250:Y4=30:S1=1:S2=-1
120 LOCATE 0,0:PRINT SC
130 IF SC=0 THEN 430
140 GOSUB 300:GOSUB 320
150 S=STICK(0):T=STRIG(0)
160 IF S=3 THEN 190
170 IF S=7 THEN 210
```

```
180 IF T=-1 THEN SC=SC-1:GOTO 240 ELSE 120
190 X1=X1+7:IF X1>230 THEN X1=230
200 GOTO 120
210 X1=X1-7:IF X1<0 THEN X1=0
220 GOTO 120
230 SPRITE ON
240 SPRITE ON:SOUND 13,1
250 FOR Y3=Y1-10 TO -25 STEP -9
260 GOSUB 320:GOSUB 410
270 ON SPRITE GOSUB 880
280 NEXT Y3
290 GOTO 120
300 PUT SPRITE 1,(X1,Y1),8,1
310 RETURN
320 PUT SPRITE 2,(X2,Y2),5,2
330 X2=X2+4*S1
340 IF X2<250 AND X2>0 THEN 360 ELSE SC=SC-1
350 IF RND(1)<.5 THEN S1=1:X2=0 ELSE S1=-1:X2=250
360 PUT SPRITE 4,(X4,Y4),5,2
370 X4=X4+4*S2
380 IF X4<250 AND X4>0 THEN 400 ELSE SC=SC-1
390 IF RND(1)<.5 THEN S2=1:X4=0 ELSE S2=-1:X4=250
400 RETURN
410 PUT SPRITE 3,(X1,Y3),15,1
420 RETURN
430 '
440 ' Fin de partie
450 CLS:SCREEN 0
460 PRINT "C'est fini..."
470 END
480 '
490 '  Decor
500 '
510 KEY OFF
520 COLOR 10,1,1:SCREEN 1,2
530 FOR I=1 TO 50
540 X=INT(RND(1)*30):Y=INT(RND(1)*20)
550 LOCATE X,Y:PRINT CHR$(42)
560 NEXT I
570 RETURN
580 '
590 ' Definition des sprites
600 '
610 S$="":FOR J=1 TO 32
620 READ C
630 S$=S$+CHR$(C)
640 NEXT J
650 SPRITE$(S)=S$
660 RETURN
670 '
680 '  Donnees des sprites
690 '
700 ' Obus
710 DATA 1,3,7,7,23,31,31,31
720 DATA 19,0,0,0,0,0,0,0
```

```
730 DATA 0,128,192,192,208,240,240,240
740 DATA 144,0,0,0,0,0,0,0
750 ' Vaisseau
760 DATA 0,63,2,4,15,16,32,64
770 DATA 255,64,32,16,15,4,2,63
780 DATA 0,248,128,128,224,16,40,68
790 DATA 132,68,40,16,224,128,128,248
800 ' Explosion
810 DATA 3,7,30,63,63,119,110,207
820 DATA 255,247,185,127,111,31,60,7
830 DATA 128,224,120,180,244,222,238,222
840 DATA 239,247,252,238,188,116,240,192
850 RETURN
860 '
870 '
880 '
890 '
900 ' Touche!
910 IF Y3<20 THEN X=X2:Y=Y2:X2=0:E=2:SC=SC+6:GOTO 930
920 X=X4:Y=Y4:X4=0:E=4:SC=SC+4
930 FOR I=1 TO 20
940 PUT SPRITE E,(X,Y),15,3
950 PUT SPRITE E,(X,Y),15,2
960 SOUND 13,0
970 NEXT I
980 SPRITE OFF
990 RETURN
```

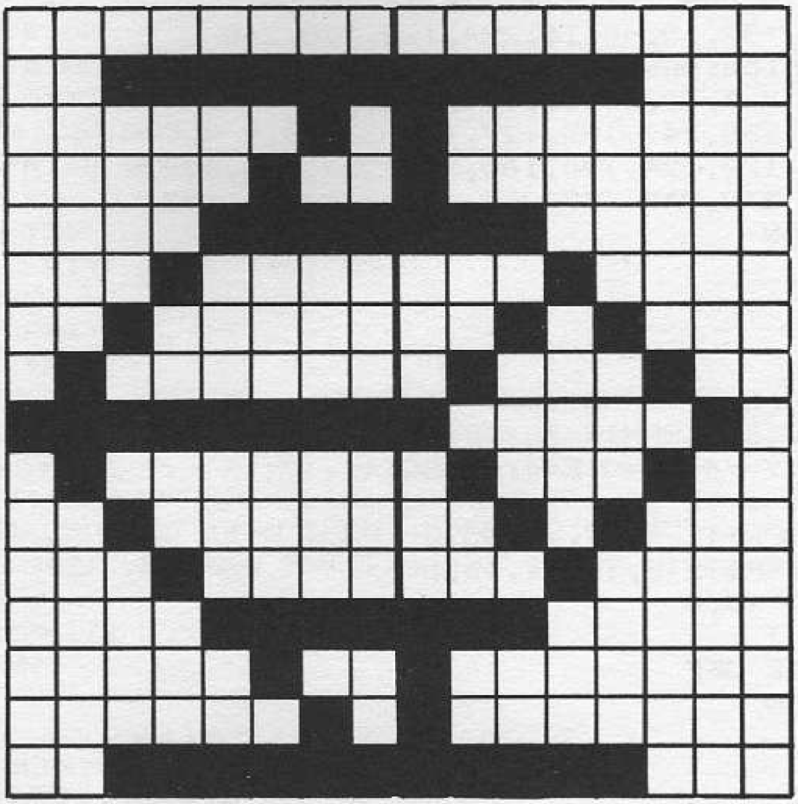
Commentaires

Nous ne détaillerons pas le programme intégralement, ce serait trop long. Nous nous contentons de commenter les sous-programmes de création et de gestion des sprites qui pourront être réinvestis ailleurs.

1. Taille
Les sprites utilisés dans le jeu sont de type double taille. Ils sont constitués de 4 caractères (16×16).
Le deuxième argument de SCREEN sélectionne la taille de définition et d'affichage des sprites. Cet argument est fixé à 2 pour des sprites 16×16 affichés en simple taille (ligne 520).

2. Données
Le programme utilise 3 sprites dont les données numériques sont en DATA à un endroit réservé (lignes 700-890).
Nous détaillons la construction du sprite appelé ici Vaisseau (lignes 750 à 800).

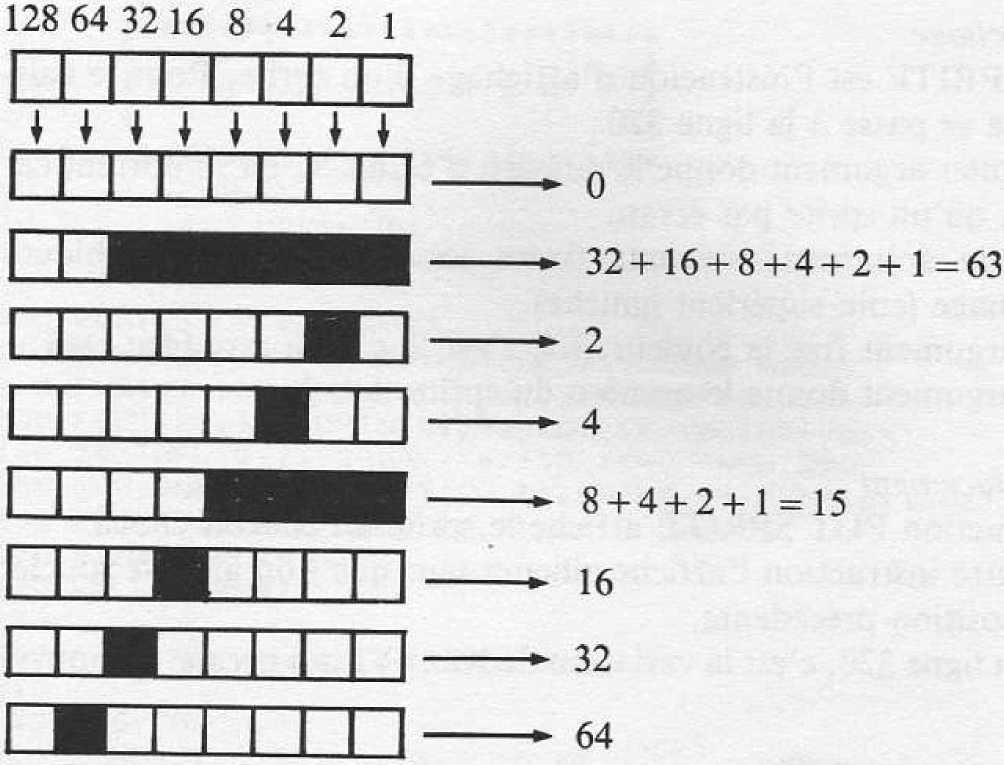
On commence par dessiner l'objet sur une grille 16×16 partagée en 4 pavés 8×8.



Pour passer de ce dessin à un codage numérique, on commence par découper le carré 16×16 en 4 carrés 8×8 dans l'ordre suivant.

1	3
2	4

Chaque carré est lui-même décomposé en 8 lignes horizontales de 8 points chacune. Chacun de ces points représente une puissance de 2 suivant sa position dans la ligne. A chaque ligne est donc associé un nombre qui caractérise la distribution des points. Voici la décomposition du 1[er] carré qui permet de retrouver la première ligne de DATA définissant le vaisseau (ligne 760).



On procède de la même façon pour les 3 autres carrés ce qui fournit les 32 données numériques définissant le sprite vaisseau.

3. Définition

C'est le sous-programme 590-660 qui se charge de définir les sprites en affectant les données convenables à la variable SPRITE$ (S). Ce sous-programme peut être utilisé ailleurs.
La construction du sprite vaisseau (Numéro 2) commence par l'installation du pointeur de données en ligne 750 où commencent les données propres au vaisseau. On fixe enfin le numéro du sprite (ici S=2). Puis on appelle le sous-programme de définition des sprites. Cet appel se fait en ligne 80.
La boucle 610-640 lit les 32 données et les charge par concaténation dans S$.
On peut remarquer que la transformation d'un nombre en un caractère se fait par la fonction CHR$. L'ensemble des données d'un sprite est donc en fait une suite de caractères du code ASCII. On peut très bien remplacer les codes par les caractères mais à moins d'avoir le code ASCII dans la tête, il est plus simple de s'en tenir au code décimal et de laisser CHR$ se charger du décodage.
L'affectation de S$ dans SPRITE$ (2) achève la définition du sprite Numéro 2.

4. Affichage

PUT SPRITE est l'instrucion d'affichage d'un sprite. Pour le vaisseau, ça se passe à la ligne 320.

Le premier argument donne le numéro d'écran. C'est important car il n'y a qu'un sprite par écran.

Les deux arguments suivants fixent les coordonnées graphiques d'affichage (coin supérieur gauche).

Le 4e argument fixe la couleur. Ici, c'est 5, c'est-à-dire bleu clair.

Le 5e argument donne le numéro du sprite. Ici, 2.

5. Déplacement

L'instruction PUT SPRITE affiche le sprite à l'endroit choisi.

Une autre instruction l'affiche ailleurs sans que l'on ait à se soucier de la position précédente.

Dans la ligne 320, c'est la variation de X2 et Y2 qui permet le mouvement.

6. Rencontre de deux sprites

Il s'agit évidemment d'un moment crucial puisque c'est celui où votre obus vengeur atteint l'ennemi tant haï.

L'activation est faite en 230 par SPRITE ON. Elle permet de dérouter le programme lors de la rencontre de deux sprites. Ce déroutement peut se faire en ligne 270 par l'instruction ON SPRITE GOSUB qui envoie sur le sous-programme 880 qui s'occupe d'animer et de bruiter l'explosion.

Miroir aux pixels

Ce jeu de symétries fera de vous un véritable artiste si vous ne l'êtes déjà.

Dessinez dans un coin d'écran et voyez dans un autre son reflet symétrique.

Sélectionnez et gommez ce qui ne vous convient pas.

Recommencez 100 fois l'ouvrage de base, l'ordinateur s'occupe du reste... L'honneur vous en reviendra tout entier.

```
10 '   ***************************
20 '   *                         *
30 '   *    S Y M E T R I E S    *
40 '   *                         *
50 '   ***************************
60 '
70 '          Choix du joueur
80 '         et initialisation
90 '
100 CLS:KEYOFF:SCREEN2
110 OPEN"GRP:" FOR OUTPUT AS #1:GOSUB 860
120 PSET(20,8):PRINT#1,"Combien d'axes de"
130 PSET(20,16):PRINT#1,"symetrie voulez-vous"
140 X$=INPUT$(1):X=VAL(X$):IF X<=0 THEN 220
150 CLS:GOSUB 860:PSET(20,8):PRINT #1,"Choisissez"
160 PSET(20,16):PRINT #1,"leurs numeros"
170 SYM$=INPUT$(X)
180 F=LEN(SYM$)
190 '
200 '      Preparation page dessin
210 '
220 CLS:COLOR,,4
230 LINE(32,0)-(224,192),15,BF
240 X=128:Y=96:Z=1:ZZ=15
250 PSET(X,Y)
260 '
270 '          Debut du dessin
280 '
290 A$=INKEY$:IF A$<>""THEN GOSUB 560
300 '
310 '          Test des limites
320 '
330 IF X<34 THEN X=33 ELSE IF X>222 THEN X=223
340 IF Y<2 THEN Y=1 ELSE IF Y>190 THEN Y=191
350 '
360 '     Test des touches flechees
370 '
380 S=STICK(0):PSET(X,Y),Z
390 ON S GOSUB 450,460,470,480,490,500,510,520
400 GOSUB 620
410 PSET(X,Y),ZZ:GOTO 290
420 '
430 '  Calcul des nouvelles coordonnees
440 '
450 Y=Y-1:PSET(X,Y),Z:RETURN
460 X=X+1:Y=Y-1:PSET(X,Y),Z:RETURN
470 X=X+1:PSET(X,Y),Z:RETURN
480 X=X+1:Y=Y+1:PSET(X,Y),Z:RETURN
490 Y=Y+1:PSET(X,Y),Z:RETURN
500 X=X-1:Y=Y+1:PSET(X,Y),Z:RETURN
510 X=X-1:PSET(X,Y),Z:RETURN
520 X=X-1:Y=Y-1:PSET(X,Y),Z:RETURN
530 '
540 '     Tracer ou Effacer ?
550 '
```

```
560 IF A$=CHR$(13) THEN SWAPZ,ZZ
570 IF A$="f" OR A$="F" THEN 950
580 A$="":RETURN
590 '
600 '  Choix des parametres pour calcul
610 '
620 FOR K= 1 TO F
630 ON VAL(MID$(SYM$,K,1)) GOTO 640,650,660,670,680
640 C1=0:C2=1:C3=1:C4=0:GOSUB 780:GOTO720
650 C1=-1:C2=0:C3=0:C4=1:GOSUB 780:GOTO720
660 C1=0:C2=-1:C3=-1:C4=0:GOSUB 780:GOTO720
670 C1=1:C2=0:C3=0:C4=-1:GOSUB 780:GOTO720
680 C1=-1:C2=0:C3=0:C4=-1:GOSUB 780
690 '
700 '     Trace du point symetrique
710 '
720 PSET(ZX,ZY),Z
730 NEXT K
740 RETURN
750 '
760 '  Calcul des coordonnees symetriques
770 '
780 ZX=X-128:ZY=Y-96
790 X1=C1*ZX+C2*ZY
800 Y1=C3*ZX+C4*ZY
810 ZX=X1+128:ZY=Y1+96
820 RETURN
830 '
840 '  Trace des lignes fictives de symetries
850 '
860 PSET(150,150):DRAW"H100R100D100L100U100"
870 PSET(150,50):DRAW"G100R50U100R50D50L100"
880 PRESET(50,40):PRINT#1,"1":PRESET(100,40)
890 PRINT#1,"2":PRESET(150,40):PRINT#1,"3"
900 PRESET(152,100):PRINT#1,"4"
910 RETURN
920 '
930 '        Fin de partie
940 '
950 PRESET(40,0):COLOR6:PRINT#1,"Dessin termine"
960 A$=INPUT$(1)
970 END
```

Commentaires

1. Le jeu commence par 2 questions
— Combien d'axes de symétrie désirez-vous sélectionner ?
— Lesquels ?
La suite est émerveillement !
Jugez-en :

2. Six touches sont à votre disposition
• les 4 touches fléchées

↑ pour aller vers le haut
→ pour aller vers la droite
↓ pour descendre
← pour virer à gauche

L'appui simultané sur :
↑ → pour se diriger en diagonale en haut et à droite (↗).
↓ → trajet en diagonale en bas et à droite (↘).
↓ ← diagonale en bas et à gauche (↙).
↑ ← diagonale en haut et à gauche (↖).

• La touche "F" (comme Fin) pour sortir du programme.

3. Un point scintillant vous indique précisément l'endroit où vous êtes.

4. Faites votre dessin, gommez, tracez, regommez, retracez... jusqu'à satisfaction. Chaque point que vous traiterez suivra le même traitement sur son (ou ses) symétrique(s).

Organisation du programme

• Une variable alphanumérique (SYM$) recueille les numéros des axes de symétrie sélectionnés (ligne 170).
• X referme le numéro de colonne du curseur.
Y repère la ligne du curseur.
Z et ZZ sont les couleurs du tracé (forme et fond).
• Le programme principal fait 7 lignes (290-410).
Il commence par un appel au sous-programme de scrutation rapide du clavier (560-580).
— Si la touche RETURN est enfoncée, les variables Z et ZZ sont permutées (instruction SWAP en ligne 560) et la couleur du tracé s'en trouve modifiée.
— Si c'est "F", on sort alors du programme principal.
• En retour de ce sous-programme, la variable STICK(0) (cf. Dictionnaire BASIC MSX) est initialisée, et oriente les calculs selon sa

valeur sur de nombreux sous-programmes imbriqués (ligne 390).
Tour à tour :
— calcul des nouvelles coordonnées X,Y (450-520) puis, dans une boucle indicée selon le nombre de symétries choisies en début de jeu ;

— sous-programme de paramétrage des calculs de symétries (620-650) ;
— sous-programme de calcul de X1,Y1 repérant les points symétriques (780-820) ;
— sous-programme de tracé ou effacement (selon la valeur de Z) du point symétrique (720-740).

Pour enfin revenir du programme principal qui gère aussi le scintillement.

• Quand on sort du programme, la variable A$ (ligne 960) attend la frappe d'un caractère au clavier.
Tant que celle-ci est vide, vous pouvez à loisir contempler le chef-d'œuvre.

Remarques

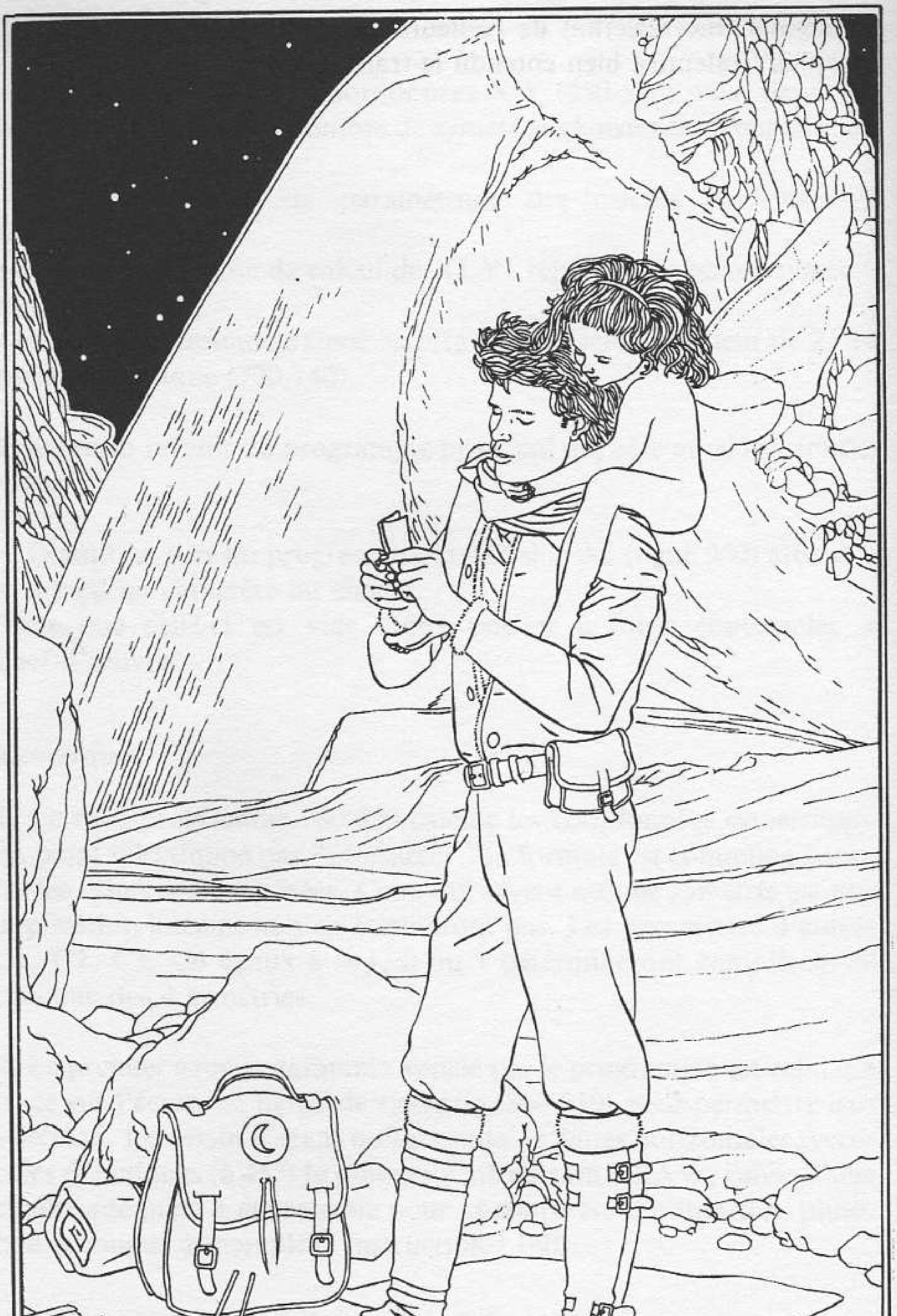
1. Le sous-programme 780-820 calcule les coordonnées symétriques du point sélectionné par l'utilisateur. La formule est commune à toutes les symétries proposées. Ceux qui savent qu'une symétrie est une application linéaire ne s'en étonneront pas. Les paramètres d'entrée C1, C2, C3, C4 égaux à −1, 0 ou 1 détermineront complètement chacune des 4 symétries.

2. Le premier sous-programme appelé par le programme est celui qui trace sur l'écran les lignes de symétrie (860-910) pour permettre leur sélection. Le dessin n'étant qu'une suite de lignes horizontales, verticales et obliques (à 45°) la puissante instruction DRAW, suivie d'une chaîne adéquate a été retenue pour sa souplesse d'utilisation, plutôt que la longue et complexe instruction LINE.

Suggestions

En maniant habilement cette technique et en étant vigilant sur la rigidité en couleur des segments, il devrait être relativement simple

d'imaginer une sélection de couleurs pour les points originaux (ce ''surplus'' ralentira bien entendu le traitement).


WALETERRE

Bloc-notes

Les programmes qui suivent sont des utilitaires qui permettent de voir ou de faire quelque chose d'intéressant. Il est utile de connaitre leur existence pour s'en servir lors du développement d'une application particulière.

— "Sprite fait, bien fait" facilite la définition des sprites.

— "Camembert" fournit une représentation graphique de répartition de nombres.

— "Mélange et tri" sont des sous-programmes standard très utiles pour les programmes éducatifs ou les programmes de jeu.

En tête

Opération terminée... Tout baigne... Et soudain, l'angoisse du créateur se noue : tant d'heures de travail devant la machine, tant d'interrogations sur l'algorithme, sur l'écriture, tant de crispations face aux bugs... tant de... pour le peu d'enthousiasme et de reconnaissance venant de ceux qui manipuleront votre programme !
"Ils" ne savent pas ce que cela représente !
Que faire pour "les" persuader de votre génie ?
Ils faut des fioritures... en voici une.

```
10 '        *****************
20 '        *               *
30 '        *    BONJOUR    *
40 '        *               *
50 '        *****************
60 '
70 '         Preparation ecran
80 '         et initialisation
90 '
100 KEYOFF:CLS:COLOR15,2,2:SCREEN0:X=10
110 PRINT"Bonjour, moi je suis MSX"
120 PRINT:PRINT:INPUT"Et vous";NOM$
130 '
140 '          Ecran graphique
150 '           et coloriage
160 '
170 CLS:SCREEN2:COLOR,,8
180 FOR J=0TO 75 STEP 15
190 LINE(0,J)-(256,J+15),J/10,BF
200 GOSUB 600
210 NEXT
220 OPEN"GRP:" FOR OUTPUT AS #1
230 PSET(20,96):PRINT#1,"Bonjour ";
240 '
250 '     Caractere par caractere
260 '
270 FOR J=1 TO LEN(NOM$)-2
280 PRINT#1, MID$(NOM$,J,1);:GOSUB 600
290 NEXT
300 '
310 '     Inversion des derniers
320 '
330 PRINT#1,RIGHT$(NOM$,1);:GOSUB 600
340 PRINT#1,MID$(NOM$,LEN(NOM$)-1,1);
350 GOSUB 600
360 FORJ=0 TO300:NEXT:FOR J=0TO4:GOSUB 600:NEXT
```

```
370 LINE(84+(8*(LEN(NOM$)-2)),96)-
    (84+(8*(LEN(NOM$))),104),2,BF
380 '
390 '          Remise en ordre
400 '
410 PRESET(84+(8*(LEN(NOM$)-2)),96)
420 PRINT#1,MID$(NOM$,LEN(NOM$)-1,1);
430 GOSUB 600:PRINT#1,RIGHT$(NOM$,1);"  "
440 GOSUB 600
450 '
460 '      On termine le coloriage
470 '
480 FOR J=108TO 159 STEP 15
490 LINE(0,J)-(256,J+15),J/10,BF
500 GOSUB 600
510 NEXT
520 '
530 '              Fin
540 '
550 A$=INPUT$(1)
560 END
570 '
580 '            Bruitage
590 '
600 FORI=0TOX:BEEP:NEXT:RETURN
```

Organisation du programme

Pas de surprise cette fois, la séquence suit pas à pas les numéros de lignes, sans saut vers des sous-programmes, excepté celui de temporisation en 600.
En 270-290, la boucle extrait et affiche caractère par caractère le contenu de NOM$ diminué de 2.
Les 2 lignes suivantes inversent l'affichage des 2 derniers caractères de cette variable.
L'expression de la ligne 370 est compliquée ; c'est une méthode servant à ''masquer'' une partie de l'écran graphique par recouvrement à l'aide de la couleur de fond. L'aridité de la formule tient au fait qu'elle est paramétrée par rapport à NOM$.
Même remarque sur la ligne 410, permettant de positionner le texte à l'écran.

Remarque

Dans les instructions LINE de traçage de boîte (190 et 490), l'argument de couleur est un calcul sur l'indice de boucle. Seule la partie

entière du calcul est prise en compte par la matrice pour détruire la teinte. Cela facilite grandement le travail et la rapidité du déroulement.

Faites sauter l'âme Morse !

Il en est qui prétendent que le morse n'est plus de mise. C'est largement vrai.
Ceci étant, il garde pour moi un aspect fascinant ; le TITITI TATATA TITITI des grandes aventures marines, le TITATI TITA TATATITI des boys-scouts...
Alors, même s'il faut reconnaître qu'a l'ère du téléphone, de la "Ci-Bi" et tutti-quanti cette façon de communiquer peut sembler bien désuète, que peut-on faire contre les souvenirs d'enfance ?
A vos claviers.

```
10 '  ******************
20 '  *                *
30 '  *      MORSE     *
40 '  *                *
50 '  ******************
60 '
70 '      Initialisation
80 '
90 PLAY"T200;L55"
100 DIM L$(26,2):CLS:KEYOFF
110 '
120 '     Chargement du tableau
130 '
140 FOR I=1 TO 26
150 FOR J=0 TO 2
160 READ L$(I,J)
170 NEXT J
180 NEXT I
190 '
200 '            Debut
210 '
220 INPUT"Entrez votre message";MES$
230 '
240 '  Decodage lettre a lettre
250 '
260 FOR I=1 TO LEN(MES$)
270 FOR J=1 TO 26
```

```
280 IF MID$(MES$,I,1)= L$(J,0) THEN
    GOSUB 350:J=26
290 NEXT J
300 NEXT I
310 END
320 '
330 '      Codage visuel
340 '
350 PRINT L$(J,1);" ";
360 '
370 '      Codage sonore
380 '
390 FOR H=1 TO LEN(L$(J,2))
400 A$= MID$(L$(J,2),H,1)
410 PLAY A$
420 NEXT H
430 PLAY "R30"
440 RETURN
450 '
460 '   Donnees du tableau
470 '
480 DATA A,.-,AD,B,-...,DAAA,C,-.-.,DADA
490 DATA D,-..,DAA,E,.,A,F,..-.,AADA,G,--.,DDA
500 DATA H,....,AAAA,I,..,AA,J,.---,ADDD
510 DATA K,-.-,DAD,L,.-..,ADAA,M,--,DD
520 DATA N,-.,DA,O,---,DDD,P,.--.,ADDA
530 DATA Q,--.-,DDAD,R,.-.,ADA,S,...,AAA
540 DATA T,-,D,U,..-,AAD,V,...-,AAAD
550 DATA W,.--,ADD,X,-..-,DAAD
560 DATA Y,-.--,DADD,Z,--..,DDAA
```

Commentaires

Le principe de cette activité est la création d'un tableau à trois dimensions et son exploitation.
Votre message, entré au clavier, vous sera répété, codé, visualisé et sonorisé.

Conseil pour la saisie

Pour plus de clarté, vous pouvez saisir les données avec une ligne particulière de Data pour chaque lettre de l'alphabet. Ainsi, vous aurez la représentation exacte du tableau en mémoire.
Par sécurité, faites-vous dicter ces données tandis que vous les frapperez au clavier.

Organisation du programme

• La ligne 100 réserve l'espace en mémoire nécessaire au stockage du tableau.
• La boucle 140-180 lit les Data et les range en tableau.
• La variable MES$, en 220, reçoit votre phrase (255 caractères au maximum).
• Le programme principal (5 lignes, 260-300), extrait en boucle, signe par signe, le contenu de MES$ et appelle le sous-programme de codage visuel et sonore.

Remarques

1. L'exploitation du sous-programme de codage utilise la valeur courante de l'indice de boucle du programme principal, tirant ainsi profit d'une ligne entière du tableau.

2. Vous pouvez ralentir l'exécution sonore en augmentant la valeur suivant R en ligne 430.

Suggestions

Il est très simple de transformer ce programme en un Jeu. Voici quelques propositions.
1. Le message entré par un tiers s'efface de l'écran.
2. Le codage sonore est émis.
3. Une aide (sous la forme du codage visuel) est fournie à la demande du joueur hésitant quand il frappe sur une touche préprogrammée.

Sprite fait bien fait

Ce n'est pas difficile, mais on y perd un temps fou ! Les puissances de 2 me font trembler et me donnent le octet.
Pourtant, sur MSX, il serait bien bête de se passer des sprites.
Alors voilà, plutôt que 256 calculs fastidieux pour mes 32 lutins, ce petit utilitaire me les dessine et calcule parfaitement.
Je vous le livre sans plus tarder.

```
10 '  **************************
20 '  *                        *
30 '  *    CALCUL D'UN SPRITE  *
40 '  *                        *
50 '  **************************
60 '
70 '          Initialisation
80 '
90 DIM CAS(8,8):X=24:Y=X
100 '
110 '     Quadrillage pour dessin
120 '            du sprite
130 '
140 CLS:KEYOFF:COLOR,4,4:SCREEN2
150 FORI=16TO144 STEP16
160 LINE(16,I)-(144,I),1
170 LINE(I,16)-(I,144),1
180 NEXT
190 '
200 '     Position du curseur
210 '
220 PSET(X,Y):GOSUB 730
230 IF POINT(X-1,Y-1)=6 THEN
    PSET(X,Y),6 ELSE PRESET(X,Y)
240 ON A GOSUB 340,350,360,370
250 IF X<24 THEN X=24
260 IF X>136 THEN X=136
270 IF Y<24 THEN Y=24
280 IF Y>136 THEN Y=136
290 GOTO 220
300 '
310 '       Nouvelle position
320 '          du curseur
330 '
340 X=X+16:RETURN
350 X=X-16:RETURN
360 Y=Y-16:RETURN
370 Y=Y+16:RETURN
380 '
390 '   Coloriage ou Effacement
```

```
400 '
410 IF POINT(X-1,Y-1)=6 THEN T=4 ELSE T=6
420 LINE(X-7,Y-7)-(X+7,Y+7),T,BF
430 I=INT((X-24)/16):J=INT((Y-24)/16)
440 ON (T/2)-1 GOTO 480,490
450 '
460 '    Mise a jour du tableau
470 '
480  CAS(I,J)=0:GOTO500
490  CAS(I,J)=1
500  RETURN
510 '
520 '        Calcul du sprite
530 '
540 FOR K=0 TO 7
550 FOR L=0 TO 7
560 IF CAS(L,K)=0 THEN 580
570 A=2^(7-L):V(K)=V(K)+A
580 NEXT L
590 PRINT
600 NEXT K
610 '
620 '       Ecriture des valeurs
630 '               et fin
640 '
650 SCREEN0:CLS:PRINT"SPRITE$(N) =":PRINT
660 FOR I=0 TO 7
670 PRINT V(I);:IF I<7 THEN PRINT",";
680 NEXT
690 END
700 '
710 '         Saisie au clavier
720 '
730 A$=""
740 A$=INKEY$:IF A$="" THEN 740
750 A=ASC(A$)
760 IF A$="F" OR A$="f" THEN 540
770 IF A=13 THEN GOSUB 410
780 IF A<28 OR A>31 THEN 730
790 A=A-27
800 RETURN
```

Commentaires

Les 4 touches fléchées du clavier permettent de déplacer à volonté un curseur dans un damier 8×8 (matrice minimum d'un sprite).
L'appui sur la touche RETURN colore la case où le curseur est présent. Si cette case était déjà "noircie", elle s'efface, laissant ainsi libre la possibilité de reprendre à satiété son dessin.

Enfin, la touche "F" (comme Fin) fait passer à une page où le sprite dessiné est codé.
Il ne reste qu'à noter ce codage et à l'intégrer à votre programme.

Organisation du programme

• En ligne 90 on réserve de la place pour un tableau à deux dimensions. Chaque case du tableau sera par la suite mise à 1 ou à 0.
• De 150 à 180, dessin du quadrillage par une boucle.
• En 220 le curseur est positionné en haut et à gauche, puis le sous-programme de test du clavier est appelé (730-800). Cette ligne à elle seule est le programme principal.
• Selon la touche frappée, ce sous-programme oriente le déroulement :
— L'une des 4 touches fléchées fait sortir de la routine et renvoie au calcul de la nouvelle position du curseur (340-370).
— RETURN saute à un second sous-programme (410-500) où est listée la couleur de la case sur laquelle on se trouve (intruction POINT), puis, fonction de ce test, la partie correspondante du damier est noircie ou blanchie.
Enfin, le tableau est mis à jour.
— L'appui sur "F" mène en fin de programme (540-600). Le tableau est lu ligne à ligne (indice K = ligne, indice L = colonne), et la valeur décimale (V(K)) calculée.
Enfin, l'affichage du calcul se fait entre 650 et 680.

Remarques

L'utilisation d'une version du sous-programme d'entrée sans Input donné par ailleurs est ici très utile : seules 6 touches du clavier sont acceptées en réponse ; toute autre réponse remonte la séquence en tête de ce sous-programme.

Il faut noter également les lignes de tests de bornes (254-280). L'échappement aux limites est bloqué.
Un tel test est recommandé sur tout programme de déplacement d'un objet ou d'un point à l'écran.

Camembert

Classique de la représentation graphique, MSX, par une instruction puissante, vous le livre fait à point.
D'un coup d'œil, les proportions qu'occupent dans votre budget l'entretien de votre voiture, vos frais de divertissement, vos impôts... ou tout autre poste de dépenses, vous seront révélées.
Alors, peut être vivrez-vous des drames familiaux... juste avant le dessert.

```
10 '     ***************
20 '     *             *
30 '     *  CAMEMBERT  *
40 '     *             *
50 '     ***************
60 '
70 '          Initialisation
80 '
90 X=125:Y=95:R=40:DEP=0:T=0:
  CO=5:PI2=6.28318
100 '
110 '
120 '
130 INPUT"Combien de donnees ";D
140 FOR I=1 TO D
150 INPUT A(I)
160 T=T+A(I)
170 NEXT
180 '
190 '         Programme principal
200 '
210 CLS:SCREEN2
220 FOR I=1 TO D
230 GOSUB 320:ARI=DEP+P:CO=CO+1:
    GOSUB 370
240 GOSUB 420:SWAP DEP,ARI
250 NEXT
260 OPEN"GRP:" FOR OUTPUT AS #1
270 PRESET(20,20):
    PRINT#1,"CAMEMBERT TERMINE"
280 A$=INPUT$(1):END
290 '
300 '          Calcul de l'angle
310 '
320 P=(PI2*(100/T*A(I)))/100
330 RETURN
340 '
350 '          Trace des arcs
```

```
360 '
370 CIRCLE(X,Y),R,CO,DEP,ARI
380 RETURN
390 '
400 '        Trace des secteurs
410 '
420 X1=X+R*SIN(DEP+1.57):
    Y1=Y+R*COS(DEP+1.57)
430 LINE(X1,Y1)-(X,Y),CO
440 X1=X+R*SIN(ARI+1.57):
    Y1=Y+R*COS(ARI+1.57)
450 LINE-(X1,Y1),CO
460 RETURN
```

Commentaires

Les quelques lignes de code qui précèdent sont à prendre comme un utilitaire, un sous-programme d'illustration de calculs effectués par ailleurs, une note "gaie" à insérer dans un dédale de nombre austères.

Organisation du programme

Le programme est constitué d'une partie principale de calculs et de deux routines graphiques rejetées en sous-programmes à la fin.
Jusqu'en ligne 170, les données brutes sont attendues par la machine et mises en tableau (150).
En ligne 160, la variable T cumule les valeurs.
De 210 à 250 le programme principal travaille en boucle, indicée à la taille du tableau, à l'appel des routines du programme :
- 320, calcul pour chaque valeur du tableau, du pourcentage relatif qu'elle occupe sur la circonférence du "camembert".
- 370, l'arc correspondant au calcul précédent est tracé.
- 420-450, le secteur attaché à cet arc se dessine.

Remarque

Les variables X et Y marquent les coordonnées du centre du cercle ; R est le rayon.
DEP et ARI sont respectivement les pointeurs de départ et d'arrivée de l'arc de cercle sur la circonférence ; ils sont mis à jour en 230 et 240.

Le "remplissage" par différentes couleurs de chacun des secteurs du "camembert" est contre-indiqué sur MSX, à cause de la "rigidité" des segments de la mémoire couleur : des effets de "bavure" et "marche d'escalier" disgracieux sont inévitables.

Mélange et tri

Mélange

Poser plusieurs questions dans le désordre en étant sûr de ne pas poser deux fois la même. C'est une démarche classique pour laquelle nous fournissons un sous-programme standard qui pourra s'adapter à toutes sortes de situations.
Supposons par exemple qu'il s'agisse de poser cinq questions parmi dix dans un ordre aléatoire. Il faut commencer par remplacer les numéros de 1 à 10 par les dix premières lettres de l'alphabet.

1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G	H	I	J

Le mélange devra opérer sur la chaîne "ABCDEFGHIJ". Pour choisir 5 caractères parmi ces 10, on procède d'une façon lumineusement naturelle. On tire au hasard un rang dans la chaîne, on extrait le caractère correspondant (MID$), puis on l'enlève de la chaîne en concaténant (+) la partie gauche (LEFT$) et la partie droite (RIGHT$). On recommence jusqu'à ce que les 5 caractères soient extraits. Il ne reste plus qu'à recoder les caractères de la chaîne obtenue en numéros de questions.

```
10 '
20 '   DEMONSTRATION MELANGE
30 '
100 CLS:PRINT"APPUYEZ SUR UNE TOUCHE"
110 Z=RND(1):IF INKEY$="" THEN 110
200 ZA=10:ZB=10
210 GOSUB 1000
220 PRINT ZB$
999 END
```

```
1000 '
1010 '      MELANGE
1020 '
1040 ZA$="":ZB$="":FOR I=1 TO ZA
1050 ZA$=ZA$+CHR$(64+I)
1060 NEXT I
1070 ZF=0
1080 ZX=1+INT(RND(1)*ZA)
1090 ZB$=ZB$+MID$(ZA$,ZX,1)
1100 ZA$=LEFT$(ZA$,ZX-1)+RIGHT$(ZA$,ZA-ZX)
1110 ZA=ZA-1:ZF=ZF+1
1120 IF ZF<ZB THEN 1080
1999 RETURN
```

Commentaires

1. Programme principal

La ligne 110 est une simulation de Randomize qui permettra d'obtenir des mélanges différents à chaque déroulement.
La ligne 200 fixe le nombre d'objets (ZB) à choisir puis à mélanger parmi ZA objets.
Ici, on a fixé ZA et ZB à 10, ce qui revient à mélanger 10 objets.

2. Sous-programmes de mélange

La boucle 1040-1060 construit une chaîne ZA$ de longueur ZA à partir du caractère A (code ASCII 65). Il s'agit d'une convention qui permettra de faire des tests de bon déroulement. Il s'agira ensuite de convertir cette chaîne en fonction de l'application.
ZF est le pointeur du nombre d'éléments tirés au hasard.
ZX est le numéro du caractère tiré dans ce qu'il reste de ZA$.
La ligne 1100 enlève le caractère tiré dans ZA$ afin qu'il ne soit pas choisi dans la suite. On concatène la chaîne des éléments à gauche et celle des éléments à droite (LEFT$ et RIGHT$).
Tant que le nombre d'éléments tirés n'atteint pas ZB, on continue le tirage.

3. Anagramme

Dans notre exemple (ZA = 10 : ZB = 10), le programme correspond à la construction d'un anagramme quelconque pour un mot de 10 lettres.

Tri

Les problèmes de tri constituent l'un des casse-têtes les plus sérieux offerts à la sagacité des informaticiens. Ne serait-ce que pour classer des nombres, du plus petit au plus grand, on pourrait y passer une vie. S'il s'agit de petits échantillons, le problème n'est pas d'un grand intérêt car quelle que soit la méthode utilisée, ça ira vite ! Par contre, s'il s'agit de plusieurs milliers de nombres, on a tout intérêt à minimiser le nombre de manipulations.

Les méthodes automatiques de tri ont peu de chances d'être "naturelles" car la machine n'a jamais un point de vue global.
Le sous-programme que nous donnons ici est un classique. La méthode utilisée est très simple pour ne pas dire simpliste. On regarde les éléments deux par deux et s'ils ne sont pas dans le bon ordre, on les permute (SWAP). On recommence ce parcours de la suite tant qu'il est nécessaire de faire des permutations. Le dernier passage ne sert à rien sinon à révéler que plus aucune permutation n'est nécessaire ; c'est dire que la suite est entièrement triée.
Cette méthode n'est performante que pour les petits échantillons.

```
10 '
20 '  DEMONSTRATION TRI
30 '
100 PRINT"COMBIEN VOULEZ-VOUS TRIER DE NOMBRES ?"
110 INPUT ZN:DIM A(ZN)
120 FOR I=1 TO ZN
130 A(I)=INT(RND(1)*100):PRINT A(I);
140 NEXT I:PRINT:PRINT
200 GOSUB 1000
210 PRINT"NOMBRES TRIES"
220 FOR I=1 TO ZN
230 PRINT A(I);
240 NEXT I
999 END
1000 '
1010 '    TRI
1020 '
1030 DR=0:FOR I=1 TO ZN-1
1040 IF A(I)<=A(I+1) THEN 1060
1050 SWAP A(I),A(I+1):DR=1
1060 NEXT I
1070 IF DR=1 THEN 1030
1999 RETURN
```

Commentaires

1. Programme principal

Le programme principal (10 à 999) a pour seul objet de tester le bon fonctionnement du sous-programme de tri (1000-1999). Il consiste à choisir ZN entiers entre 1 et 100, à les afficher puis à dérouter sur le sous-programme avant de réafficher la suite triée. On pourra constater de cette façon, qu'en augmentant ZN régulièrement, le temps du tri augmente avec ZN, ce qui n'est pas étonnant mais que cette progression est de type exponentiel.
Ainsi, pour trier 100 nombres, il faut environ une minute et demi ce qui prouve bien que la méthode utilisée n'est pas la plus performante qu'on puisse imaginer.

2. Sous-programme de tri

- Un drapeau DR est mis à 0 en 1030.
- Dans la boucle 1040-1060, on teste si deux nombres consécutifs A(I) et A(I + 1) sont dans le bon ordre. Lorsque ce n'est pas le cas, on les échange par l'instruction SWAP et on lève le drapeau (ligne 1050).
- Tant que le drapeau est levé, il faut continuer (ligne 1070).
- Lorsque toute la suite est parcourue sans qu'aucune permutation n'ait été effectuée, c'est que la suite est totalement triée.


WALETERRE

Dictionnaire Basic MSX

ABS
Donne la valeur absolue d'une expression numérique placée entre parenthèses. Le type du résultat est le même que celui de l'argument.

ABS (-5.38) = 5.38
ABS (15) = 15

AND
ET logique. Placé entre deux expressions, il donne une valeur numérique correspondant à vrai ou faux suivant la table d'opération suivante.

AND	Ø	1
Ø	Ø	Ø
1	Ø	1

Ø = Faux
1 = Vrai

ASC
Donne le code ASCII du premier caractère de la chaîne placée entre parenthèses. Cette chaîne ne doit pas être vide.

ASC ("A") = 65
ASC ("BABA") = 66

ATN
Donne l'angle en radian (entre – PI/2 et PI/2) dont la tangente (cf. TAN) est donnée entre parenthèses.

```
ATN(0)=0
ATN(-1)=-.78539816339745
```

AUTO
Commande de numérotation automatique de lignes. AUTO100,5 fournira automatiquement les lignes 100, 105, 110, 115, etc. On en sort par CONTROL/STOP.

BASE
Renvoie l'adresse mémoire de début des données servant à l'affichage. Ne s'utilise que dans les programmes en langage machine.

BEEP
Émet un son standard. Équivalent de PRINT CHR$(7).

BIN$
Convertit un nombre décimal en une chaîne de caractères contenant la décomposition binaire du nombre.

```
BIN$(65)="1000001"
```

BLOAD
Charge en mémoire centrale un programme en langage machine stocké sur cassette (cf. BSAVE).

BSAVE
Stocke une zone mémoire sur cassette. BSAVE doit être suivie du nom choisi pour le fichier, de l'adresse de début de la zone sauvegardée, de l'adresse de fin, puis de l'adresse de lancement s'il s'agit d'un programme exécutable.

CALL SYSTEM
Commande la sortie du Basic et le passage sous MSX-DOS.

CDBL
Convertit un nombre en double précision.
X# = CDBL(X) est une instruction qui transfère le contenu d'une variable simple précision (X) en une variable double précision X#.

CHR$
Donne le caractère correspondant à la valeur du code ASCII placé entre parenthèses (nombre entier entre 0 et 255).

```
CHR$ (65) = "A"
CHR$ (55) = "7"
```

Les codes entre 0 et 31 sont des caractères de contrôle. **ntrôle.**

CINT
Convertit un nombre en type entier.
X% = CINT(X) est une instruction qui transfère le contenu de la variable X en une variable entière X% et en supprimant éventuellement la partie décimale de X.

CIRCLE
Dessine un cercle, un arc ou une ellipse.

- CIRCLE(50,50),30,6 trace un cercle rouge (code 6) centré au point de coordonnées (50,50) avec un rayon de 30.
- CIRCLE(50,50),30,6,0,PI/2 trace un arc de cercle entre 0 et 45° (sens trigonométrique).
- CIRCLE(50,50),30,6,0,2*PI,3 trace une ellipse dont l'axe vertical est 3 fois plus grand que l'axe horizontal.
- CIRCLE STEP(20,10),30 trace un cercle de rayon 30 dont le centre est décalé horizontalement de 20 et verticalement de 10 par rapport à la position courante du curseur graphique.

CLEAR
Cet ordre réinitialise les variables et permet de réserver de la place en mémoire pour le programme.
Le premier argument réserve de la place pour les chaînes de caractères, le deuxième argument précise l'adresse maximum utilisable par l'interpréteur Basic.

CLOAD
Commande de chargement en mémoire vive d'un programme stocké en mémoire de masse (cassette ou disquette).
LOAD est suivi du nom sous lequel le programme a été stocké en mémoire de masse (voir SAVE) placé entre guillemets.

CLOAD?
Vérifie le bon enregistrement d'un programme Basic.
En cas d'erreur, un message est affiché.

CLOSE
Ferme un fichier ouvert par OPEN en interrompant la communication entre l'unité centrale et le canal de lecture/écriture.

CLS
Efface tous les caractères et graphiques affichés. Cette instruction vide donc la mémoire d'écran et positionne le curseur en haut et à gauche. Le même résultat peut être obtenu par

```
PRINT CHR$ (12).
```

COLOR
Définit la couleur pour l'écriture, le fond et le bord d'écran.
code des couleurs :

Ø	Transparent	8	Rouge moyen
1	Noir	9	Rouge clair
2	Vert moyen	10	Jaune foncé
3	Vert clair	11	Jaune clair
4	Bleu foncé	12	Vert foncé
5	Bleu clair	13	Magenta
6	Rouge foncé	14	Gris
7	Cyan	15	Blanc

• COLOR10,4,8 définit un écran BLEU (4) bordé de Rouge moyen (8) sur lequel l'écriture sera Jaune foncé (10).

CONT
Relance un programme interrompu par l'instruction STOP ou au clavier (CONTROL/STOP) à l'endroit de l'interruption et sans modification des variables.

COS
Donne le cosinus d'un angle exprimé en radians.

```
COS (0) = 1
COS (3.14) = -1
```

CSAVE
Commande de transfert en mémoire de masse du programme présent en mémoire centrale (mémoire vive).
SAVE doit être suivi d'un nom entre guillemets sous lequel le programme sera stocké en mémoire de masse (cassette) cf. CLOAD.

CSNG
Convertit un nombre en simple précision (cf. CDBL et CINT).

CSRLIN
Donne la position verticale du curseur (entre 0 et 23) cf. POS. Après l'instruction LOCATE 8,10,CSRLIN vaut 10.

DATA
Cette instruction permet d'incorporer au programme une liste de données (constantes) qui pourront être lues au moyen d'une instruction READ.

DEFDBL
Instruction de déclaration de type de variable en double précision (en début de programme). Une tranche d'alphabet est déclarée derrière DEF DBL de telle façon que toutes les variables numériques dont la première lettre appartient à la tranche définie sont automatiquement de type réel double précision.
Ex : l'instruction DEFBL A-C entraîne que toutes les variables dont le nom commence par A, B ou C sont réelles en double précision.

DEFFN
Définit une fonction agissant sur la ou les variables indiquées entre parenthèses. Les fonctions ainsi définies s'appliquent aussi bien aux variables numériques qu'aux variables chaînes de caractères et peuvent être utilisées comme les fonctions ordinaires du Basic.
Exemple : **fonction distance**
Si : DEFFN DIST (X,Y) = SQR(X*X + Y*Y)
alors : FN DIST (3,4) = 5

DEFINT
Instruction de déclaration de type variable entière. Une tranche d'alphabet est déclarée derrière DEFINT de telle façon que toutes les variables numériques dont la première lettre appartient à la tranche définie sont automatiquement de type entier.
Ex : l'instruction DEFINT H-K entraîne que toutes les variables dont le nom commence par H, I, J ou K sont entières.

DEFSNG
Instruction de déclaration de type variable en simple précision.

DEFSTR
Déclaration de type chaîne de caractères. Pour toute variable ainsi déclarée, le suffixe $ devient inutile.

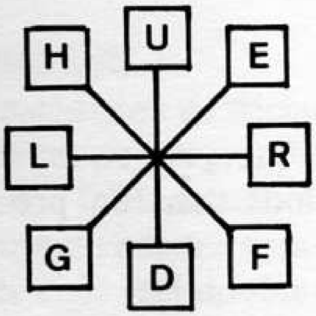
DEFUSR
Déclaration de stockage d'adresse d'une routine en langage machine (10 routines maximum).
DEFUSR0 = 60000
permet de définir l'adresse de lancement (60000) de la routine 0. cf. USR.

DELETE
Commande de suppression de lignes d'instructions dans un programme.
DELETE 10 : supprime la ligne 10.
DELETE 10-30 : supprime toutes les lignes de 10 à 30 (bornes comprises).
DELETE 10- : supprime toutes les lignes à partir de 10.

DIM
Instruction de dimensionnement de tableau. Elle réserve la place demandée en mémoire pour des tableaux à une ou plusieurs dimensions. En général, toutes les instructions DIM sont placées en début de programme. Il est possible de dimensionner plusieurs fois le même tableau en utilisant l'instruction ERASE.
DIM A(15) : réserve 15 places dans le tableau de variables numériques A.
DIM M$(5,10) : réserve 50 places dans le tableau de variables chaînes M$ (deux dimensions).

DRAW
Instruction lançant l'exécution d'une séquence graphique codée dans une chaîne de caractères.
Le principe du codage utilise la notion de déplacement relatif à partir du dernier point dessiné. Chaque lettre de la chaîne de caractères est suivi d'un ou deux argument numérique qui fixent l'amplitude du déplacement.

Codage des lettres :
U : déplacement vers le haut (up)
D : déplacement vers le bas (down)
L : déplacement vers la gauche (left)
R : déplacement vers la droite (right)
E : déplacement en diagonale haut-droit
F : déplacement en diagonale bas-droit
G : déplacement en diagonale bas-gauche
H : déplacement en diagonale haut-gauche



M : déplacement vers un point dont les coordonnées sont indiquées. Si les deux arguments sont sans signe, les coordonnées sont considérées comme absolues ; si les deux arguments sont précédées du signe + ou −, les coordonnées sont considérées comme relatives.
M100,80 trace une ligne du point courant jusqu'au point de coordonnées (100,80)
M50, − 40 trace une ligne du point courant jusqu'au point obtenu en se déplaçant de 50 à droite et de 40 en bas.
B : agit comme M sans laisser de trace.
N : ramène le curseur à la dernière position sans laisser de trace.
A: concaténé à une chaîne, permet d'exécuter le dessin avec une rotation.
A0 : rotation 0°
A1 : rotation 90° (sens trigonométrique)
A2 : rotation 180°
A3 : rotation 270°
C : définit la couleur du tracé. C doit être suivi d'un code couleur (de 0 à 15)
S : définit un facteur d'échelle. S doit être suivi d'un nombre n entre 1 et 255. Le facteur d'échelle est le quart de n.
X : exécute une sous-chaîne à l'intérieur de la chaîne donnée en argument de DRAW. Pour exécuter B$ dans A$, on écrira X B$; dans A$.

ELSE
cf. IF

END
Instruction de fin d'exécution d'un programme. En général, elle est la dernière instruction du programme principal et précède la première ligne du premier sous-programme.
L'instruction END ne modifie pas le contenu des variables.

EOF
Indique la fin d'un fichier séquentiel (End of File).
EOF vaut (– 1) si la fin du fichier est atteinte, 0 dans le cas contraire.

EQV
Opérateur logique d'équivalence. Placé entre deux expressions, il donne une valeur numérique correspondant à vrai ou faux suivant la table d'opération :

EQV	Ø	1
Ø	1	Ø
1	0	1

Ø : Faux
1 : Vrai

ERASE
Élimine un ou plusieurs tableaux de la mémoire.
ERASE A( ), B$(,) efface les tableaux A et B$ qui pourront alors être redimensionnés par DIM.

ERL
Donne le numéro de la ligne où une erreur s'est produite.
cf. ON ERROR GOTO, ERR, RESUME, ERROR

ERR
Donne le code de l'erreur.
cf. ERL, ON ERROR GOTO, RESUME, ERROR

ERROR

Permet de définir des codes d'erreur personnalisés. Les codes d'erreur du Basic sont entre 0 et 70.

• ERROR 15 simule cette erreur (interruption du programme et affichage du message correspondant).

• ERROR 100 créé une erreur de code 100.

cf. ERL, ON ERROR GOTO, RESUME, ERR.

EXP

Exponentielle calculée à partir de la constante
e = 2.71828.

EXP(0.5) = 1.6487212707
EXP(2) = 7.38905609893

FIX

Donne la partie d'un nombre située à gauche de la virgule.

FIX(4.56) = 4
FIX(− 2.8) = − 2
cf. ABS

FOR, TO, NEXT

Instructions de boucle. La variable qui suit FOR prend toutes les valeurs entières encadrant TO. Toutes les instructions entre FOR et NEXT sont exécutées à chaque passage.

A toute instruction FOR doit correspondre une instruction NEXT et une seule.

Exemple :

```
FOR X=5Ø TO 1ØØ
  ⋮          Instructions
NEXT X
```

Plusieurs boucles peuvent être emboîtées. Les positions respectives des FOR - NEXT correspondant à chacune de ces boucles doivent impérativement respecter les règles d'emboîtement.

Il faut entrer dans les boucles par FOR et en sortir par NEXT. Toute sortie prématurée provoque un encombrement de la pile.

FRE
Fonction donnant la taille mémoire disponible, en octets.
FRE(Ø) donne la taille disponible pour le programme et les variables.
FRE(X$) où X$ est une chaîne quelconque, donne la taille disponible pour le traitement des chaînes de caractères.

GOSUB
Instruction de branchement vers un sous-programme.
GOSUB est suivi d'un numéro de ligne qui est la première ligne du sous-programme appelé. Après exécution du sous-programme, l'instruction RETURN ramène le déroulement à la première instruction suivant GOSUB.
Un sous-programme peut s'appeler lui-même (appels récursifs), le programmeur doit contrôler lui-même les variables qui permettront de sortir du sous-programme.
Exemple : **calcul de factorielle N**

```
10 INPUT N
20 FACTN=1 : GOSUB 100
30 PRINT FACTN
40 END
100 REM
110 IF N=1 THEN GOTO 199
120 FACTN=FACTN*N : N=N-1 : GOSUB 100
199 RETURN
```

GOTO
Instruction de branchement sur le numéro de ligne suivant GOTO.
Le déroulement se poursuit alors à partir de cette ligne.

HEX$
Convertit un nombre décimal en une chaîne de caractères contenant la traduction hexadécimale du nombre.

HEX$(65535)="FFFF"

IF,THEN,ELSE
Instruction de branchement conditionnel selon les résultats d'un test.

IF test THEN instruction 1 ELSE instruction 2

Dans les instructions suivant THEN et ELSE, s'il s'agit d'une seule instruction de branchement sur une ligne, on peut omettre GOTO.

IF A=B THEN 100 ELSE 200

branche en ligne 100 si A est égal à B, en ligne 200 dans le cas contraire.

IMP
Opérateur logique d'implication. Placé entre deux expressions, il donne une valeur numérique correspondant à vrai ou faux suivant la table d'opération :

IMP	Ø	1
Ø	1	Ø
1	1	1

Ø : Faux
1 : Vrai

Cette opération n'est pas commutative.

INKEY$
Fonction renvoyant le caractère frappé au clavier au moment du passage sur cette instruction. Le caractère peut être stocké dans une variable chaîne et traité normalement. Si aucun caractère n'est frappé, INKEY$ reste vide.
Exemple : **attente d'un caractère.**
10 A$=INKEY$: IF A$="" THEN GOTO 10

INP
Donne l'octet lu sur un port.
INP(10) renvoie l'octet du port 10.

INPUT
Instruction de lecture du clavier. Elle provoque l'affichage d'un ? et attend les caractères frappés (maximum 255) qui sont rangés, après

appui sur la touche ENTRÉE, en mémoire et sous le nom de variable indiqué derrière INPUT.
Les erreurs de type, provoquent le message "Redo" et un nouveau passage sous INPUT. Lorsque la réponse est acceptée, le programme continue en séquence.
Plusieurs données peuvent être enregistrées sous un seul INPUT. Les noms des variables sont alors séparés par une virgule et l'utilisateur devra faire de même en entrant les données.

INPUT #
Instruction de lecture d'une donnée sur un fichier.
INPUT # doit être suivi d'un numéro de fichier et d'une liste de variables auxquelles les données lues seront affectées.
Toute instruction INPUT # doit être précédée d'une instruction OPEN d'ouverture et suivie d'une instruction CLOSE de fermeture.
cf. CLOSE, OPEN, PRINT #, EOF

INPUT$
Saisie d'un nombre déterminé de caractères sans affichage à l'écran.
INPUT$ (3) attend la saisie d'exactement 3 caractères.

INSTR
Fonction de recherche d'une sous-chaîne dans une chaîne. INSTR X$, Y$ renvoie un entier correspondant au rang à partir duquel la chaîne Y$ a été rencontrée dans X$. Si la chaîne n'est pas rencontrée, la fonction renvoie Ø.

```
INSTR ("REGRETTABLE","TABLE") = 7
INSTR ("BASIC","F") = 0
```

INT
Fonction donnant le plus petit entier inférieur ou égal au nombre donné entre parenthèses.

```
INT (5.7) = 5
INT (-3.6) = -4
```

INTERVAL ON/OFF/STOP
Active, désactive ou inhibe un branchement déclaré par ON INTERVAL GOSUB.
• INTERVAL ON : active le branchement en effectuant automatiquement un contrôle du temps écoulé à partir de cette instruction.
• INTERVAL OFF : désactive le branchement.
• INTERVAL STOP : inhibe le branchement jusqu'à rencontre de l'instruction INTERVALON.
cf. ON INTERVAL, STOP, ON STOP

KEY
Affecte à une touche fonction, une chaîne de moins de 15 caractères.
KEY2,"FIN"
affecte la chaîne "FIN" à la touche de fonction N°2.
Cf. ON KEY, KEY ON, KEY OFF, KEY LIST

KEY LIST
Commande le listage à l'écran des chaînes de caractères associées aux dix touches de fonction.
cf. : KEY, ON KEY, KEY ON, KEY OFF

KEY ON
Rétablit l'affichage des chaînes de caractères associées à 5 touches de fonction (N° 1 à 5 sans SHIFT, N° 6 à 10 avec SHIFT). La 24e ligne d'écran est réservée à cet affichage.
cf. : KEY, ON KEY, KEY LIST, KEY OFF

KEY OFF
Supprime l'affichage des chaînes de caractères associées aux touches fonction et libère la 24e ligne d'écran.
cf. : KEY, ON KEY, KEY LIST, KEY ON

LEFT$
Fonction donnant la partie gauche d'une chaîne de caractères. LEFT$ (X$, I) renvoie les I premiers caractères de X$.

Si I est nul, la chaîne renvoyée est vide. Si I dépasse la longueur de X$, X$ est renvoyé.

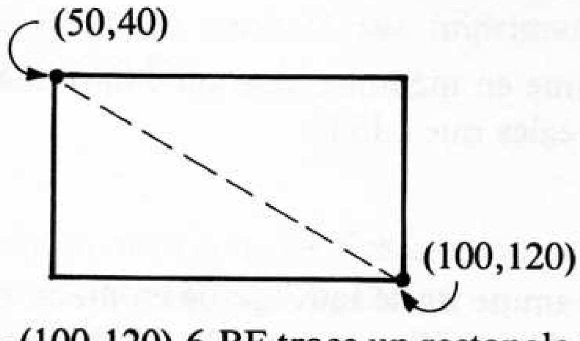
LEFT$ ("ABCD",3) = "ABC"
LEFT$ ("ABCD",0) = ""
LEFT$ ("ABCD",5) = "ABCD"

LEN
Fonction donnant la longueur de la chaîne de caractères donnée entre parenthèses.

LEN ("ABCD") = 4
LEN ("") = 0

LET
Instruction d'affectation permettant de ranger une valeur sous un nom de variable. En Basic MSX, LET est optionnel.

LINE
Instruction graphique permettant de tracer, une ligne, un rectangle, un rectangle plein.
• LINE (50,40) – (100,120),6 trace une ligne rouge (code 6) entre le point de coordonnées (50,40) et le point de coordonnées (100,120).
• LINE (50,40) – (100,120),6,B trace un rectangle rouge dont deux sommets opposés ont pour coordonnées (50,40) et (100,120). B est mis pour Box.



• LINE (50,40) – (100,120),6,BF trace un rectangle plein. BF est mis par Box Fill.
• LINE – (80,100) trace dans la couleur courante, une ligne allant de la position courante du curseur graphique jusqu'au point de coordonnées (80,100). Fonctionne de la même façon avec les arguments B ou BF.

• LINE STEP (20, – 10) – STEP(– 50,40) trace une ligne dont les extrémités sont fixées par les couples suivant STEP compris comme des déplacements relatifs par rapport à la position du curseur graphique. Ici, si le curseur graphique est en (X,Y), la ligne ira de (X + 20,Y – 10) jusqu'à (X – 50,Y + 40). Fonctionne de la même façon avec les arguments B et BF.

LINE INPUT
Instruction de lecture au clavier d'une chaîne de caractères sans affichage du point d'interrogation (voir INPUT).
Le séparateur virgule est autorisé dans la chaîne.

LINE INPUT #
Instruction de lecture dans un fichier enregistré sur cassette avec les mêmes tolérances que LINE INPUT (possibilité d'indice des séparateurs).

LIST
Commande d'affichage de lignes d'instructions d'un programme en mémoire.
LIST 10 : affiche la ligne 10.
LIST 10 - 30 : affiche toutes les lignes de 10 à 30 (bornes comprises).
LIST 10 - : affiche toutes les lignes à partir de 10.
LIST : affiche toutes les lignes du programme.

LLIST
Liste le programme en mémoire vive sur l'imprimante.
Suit les mêmes règles que LIST.

LOAD
Charge un programme Basic sauvegardé en mode ASCII par SAVE.
Il est nécessaire de spécifier le nom du périphérique (CAS pour la cassette).
LOAD"CAS:TRUC",R
R provoque le lancement automatique du programme à la fin du chargement.
cf. : SAVE, MERGE

LOCATE
Fixe sur l'écran la position et la présence du curseur d'écriture.
LOCATE 15,10,1 : fait apparaître le curseur colonne 15, ligne 10.
LOCATE 15,10,0 : place le curseur colonne 15, ligne 10 sans écho à l'écran.
LOCATE 10 : place le curseur colonne 10 dans la ligne courante.
LOCATE,15 : place le curseur ligne 15 dans la colonne courante.
Les 40 colonnes sont numérotés de 0 à 39 et les 24 lignes de 0 à 23.

LOG
Fonction donnant le logarithme nepérien d'un nombre positif.

LOG(5) = 1.6094379124341

LPOS
Donne la position de la tête d'impression dans le buffer de l'imprimante (emplacement de la prochaine impression) LPOS doit être suivi d'un argument numérique fictif (ex. : LPOS(1))
cf. : POS, LPRINT.

LPRINT
Instruction d'affichage sur l'imprimante. Suit les mêmes règles que PRINT.

LPRINTUSING
Instruction d'affichage formaté sur imprimante. Suit les mêmes règles que PRINTUSING.

MAXFILES
Spécifie le nombre maximum de fichiers pouvant être ouverts simultanément en mémoire (entre 0 et 15).
cf. : OPEN, CLOSE.

MERGE
Fusionne le programme résident en RAM avec un programme sauvegardé en mode ASCII sur la cassette. Les lignes du programme

appelé s'intercalent ou recouvrent celles du programme en RAM.
cf. : SAVE, LOAD.

MID$

Fonction permettant d'extraire une partie d'une chaîne de caractères. MID$ (X$, I, J) renvoie une chaîne formée des J caractères de X$ à partir du rang I.

```
MID$ ("ABCD", 2, 1)="B"
```

MOD

Opération donnant le reste de la division euclidienne entre deux opérandes de type entier.

```
6 MOD 2=0
17 MOD 5=2
```

MOTOR ON/OFF

Commande le moteur du magnétocassette équipé d'une télécommande (Remote)
MOTORON : lance le moteur
MOTOROFF : arrête le moteur
MOTOR : lance ou arrête suivant l'état du moteur (flip-flop).

NEW

Commande d'effacement du programme présent en mémoire centrale.

NEXT

Cf. : FOR.

NOT

Fonction logique transformant le vrai en faux et le faux en vrai.

NOT	Ø	1
	1	Ø

Ø : Faux
1 : Vrai

OCT$
Convertit un nombre décimal en une chaîne de caractères contenant la décomposition octale du nombre.

```
OCT$(169)="251"
```

ON ERROR GOTO
Instruction de branchement en cas d'erreur dans le déroulement du programme.
ON ERROR GOTO 1000 placé en début de programme déroutera vers la ligne 1000 en cas d'erreur.
cf. : ERL, ERR.

ON, GOSUB
Instruction de branchement conditionnel sur un sous-programme suivant la valeur prise par une variable entière. ON K GOSUB 1000, 2000, 3000 branche sur les sous-programmes 1000, 2000 et 3000 suivant que K vaut 1, 2 ou 3. Si K dépasse les valeurs permises, l'instruction de branchement est ignorée.

ON, GOTO
Instruction de branchement conditionnel sur une ligne d'instructions suivant la valeur prise par une variable entière.
ON K GOTO 100, 200, 300 branche sur les lignes 100, 200 ou 300 suivant que K vaut 1, 2 ou 3. Si K dépasse les valeurs permises, l'instruction est ignorée.

ON, INTERVAL, GOSUB
A partir de la rencontre d'une telle instruction, le programme se branchera automatiquement à intervalles de temps réguliers sur un sous-programme.
ON INTERVAL=500 GOSUB 1000 déroute le programme sur le sous-programme 1000 toutes les 10 secondes (1 seconde correspond environ à 50).
INTERVAL ON est nécessaire pour activer le branchement, INTERVAL OFF pour le désactiver et INTERVAL STOP pour l'inhiber.

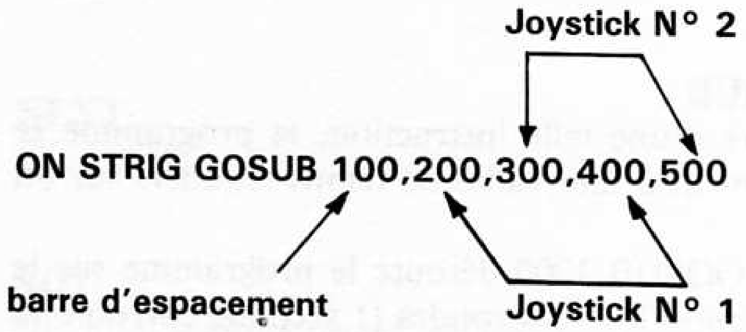
ON KEY GOSUB
Branche sur un sous-programme associé aux touches de fonction.
ON KEY GOSUB 100, 200, 300 branche sur le sous programme 100 si l'utilisateur appuie sur la touche fonction 1, 200 pour la touche fonction 2 et 300 pour la touche fonction 3.

ON SPRITE GOSUB
Branche sur un sous-programme en cas de rencontre entre deux sprites.
ON SPRITE GOSUB 1000 déroute le programme sur le sous-programme 1000 dès la collision de deux sprites.
SPRITE ON active l'instruction de branchement, SPRITE OFF la désactive et SPRITE STOP l'inhibe.

ON STOP GOSUB
Instruction de branchement permettant de contrôler une interruption éventuelle de l'utilisateur par CONTROL/STOP.
ON STOP GOSUB 1000 provoque un branchement au sous-programme 1000 dès que l'utilisateur appuie sur CONTROL/STOP.

STOP ON active l'instruction de branchement, STOP OFF la désactive et STOP STOP l'inhibe.

ON STRIG GOSUB
Instruction de branchement contrôlant l'état de la barre d'espacement du clavier et des touches de deux joysticks.



cf. : STRIG, STICK

OPEN FOR, AS
Commande d'ouverture de fichier. Derrière OPEN, doivent être indiqués dans l'ordre :
— Le périphérique.
CRT : pour l'écran alphanumérique
GRP : pour l'écran graphique
LPT : pour l'imprimante
CAS : pour la cassette
— Le nom du fichier (facultatif)
— Le mode
OUTPUT pour l'écriture
INPUT pour la lecture
APPEND pour ajouter en écriture des données en fin de fichier
— Le numéro du fichier précédé d'un #
• OPEN "GRP:"FOR OUTPUT AS #1 ouvre le fichier 1 d'écriture sur l'écran graphique.
• OPEN "LPT:"FOR OUTPUT AS #1 ouvre le fichier 1 d'écriture sur l'imprimante.
• OPEN "CAS:TRUC"FOR APPEND AS #1 ouvre le fichier TRUC en écriture pour y ajouter de nouvelles données.
cf. : PRINT#, INPUT#, LINEINPUT#, PRINTUSING#, CLOSE

OR
Ou logique. Cette opération placée entre deux expressions donne une valeur numérique correspondant à vrai ou faux suivant la table d'opération :

OR	Ø	1
Ø	Ø	1
1	1	1

Ø = Faux
1 = Vrai

PAD
Donne l'état d'un périphérique de type paddle.
• PAD(0) ou PAD(4) vaut (– 1) si l'une des 8 touches du paddle 1 ou du paddle 2 est appuyée (0 sinon)
• PAD(1) ou PAD(5) donne la valeur de l'abscisse du point désigné par le paddle 1 ou le paddle 2.

• PAD(2) ou PAD(6) donne la valeur de l'ordonnée du point désigné par le paddle 1 ou le paddle 2.
• PAD(3) ou PAD(7) vaut (– 1) si l'interrupteur du paddle 1 ou du paddle 2 est appuyé (0 sinon)
• 0,1,2,3 concernent le paddle 1
4,5,6,7 concernent le paddle 2
• Les coordonnées (1,2 et 5,6) ne sont données qu'après activation des touches et interrupteur (0,3 et 4,7)

PAINT
Instruction graphique de remplissage d'un domaine de l'écran limité par une courbe fermée.
• PAINT (50,50),6 remplit en rouge la zone limitée par une courbe fermée rouge et contenant le point de coordonnées (50,50)
• PAINT STEP (– 10,10),6 remplit en rouge la zone limitée par une courbe fermée rouge et contenant le point situé à 10 pixels à gauche et 10 pixels en haut de la position du curseur graphique courant (déplacement relatif).
• PAINT (10,10),6,4 est une instruction de remplissage en mode basse résolution. Le deuxième code couleur (ici 4) sert à définir la couleur attendue du pourtour.
En haute résolution (écran graphique) la couleur de remplissage est nécessairement identique à la couleur du bord.

PDL
Décrit l'état d'une périphérique connecté sur les ports JOYSTICK.
cf. : PAD

PEEK
Donne le contenu de l'adresse mémoire dont le numéro est indiqué entre parenthèses. PEEK (163Ø5) donne le contenu de la case mémoire d'adresse 163Ø5, c'est-à-dire un octet dont la valeur décimale est comprise entre Ø et 255.

PLAY
Emet sur les 3 canaux une mélodie codée dans 3 chaînes de caractères.

• PLAY A$,B$,C$
joue simultanément la chaîne A$ sur le canal 1, B$ sur le canal 2 et C$ sur le canal 3.
• Codage des chaînes :
A,B,C,D,E,F,G désignent les 7 notes de la gamme. Le signe # ou + est mis pour les dièses, – pour les bémols.
O désigne l'octave. O est suivi d'un entier entre 1 et 8. 1 pour l'octave la plus basse, 8 pour l'octave la plus haute. Par défaut, l'octave est 4.

N désigne la hauteur de la note. N est suivi d'un entier entre 0 et 96 correspondant aux 8 octaves. La correspondance avec les notes de la gamme est donnée par le tableau suivant.

Note \ Octave	1	2	3	4	5	6	7	8
Do . C	1	13	25	37	49	61	73	85
Do# . C#	2	14	26	38	50	62	74	86
Ré . D	3	15	27	39	51	63	75	87
Ré# . D#	4	16	28	40	52	64	76	88
Mi . E	5	17	29	41	53	65	77	89
Fa . F	6	18	30	42	54	66	78	90
Fa# . F#	7	19	31	43	55	67	79	91
Sol . G	8	20	32	44	56	68	80	92
Sol# . G#	9	21	33	45	57	69	81	93
La . A	10	22	34	46	58	70	82	94
La# . A#	11	23	35	47	59	71	83	95
Si . B	12	24	36	48	60	72	84	96

0 correspond à un silence.
Ce système facilite les changements de tonalité.

L désigne la longueur de la note émise. L est suivi d'un entier entre 1 et 64. : 1 pour la note la plus longue et 64 pour la plus courte. Par défaut la longueur est 4. L8 donne une note 2 fois plus courte et L2 deux fois plus longue.

L fixe la longueur des notes émises jusqu'à la nouvelle instruction de ce type. Pour changer la longueur d'une seule note, on fait suivre son nom (A... G) de la longueur désirée, et la longueur courante reste valable pour les autres.

Ex. : L8 A16 BC

BC → B de longueur 8, C de longueur 8

A16 → A de longueur 16

L8 → longueur courante 8

R émet un silence. L'entier suivant R indique la longueur du silence entre 1 et 64 (par défaut : 4).

T désigne le tempo (nombre de noires à la minute). T est suivi d'un entier entre 32 et 255 (par défaut : 120).

. permet de pointer une note (multiplication de la longueur par 3/2)

V règle le volume de sortie. V est suivi d'un entier entre 0 et 15 (par défaut : 8). 0 correspond au silence.

M règle la fréquence de l'enveloppe entre 1 et 65535 (cf. SOUND) (par défaut : 255).

S sélectionne l'une des 8 formes d'onde (par défaut : 1). cf. SOUND.

X permet d'exécuter une sous-chaîne à l'intérieur d'une chaîne. Pour exécuter A$, on écrit XA$;

PLAY(1), PLAY(2) et PLAY(3) renseignent sur l'état des trois canaux et vaut (– 1) si la mélodie est en cours d'interprétation dans le canal, 0 si elle est terminée.
PLAY(0) vaut (– 1) si l'un au moins des trois canaux est actif.

POINT
Renvoie un code indiquant la couleur du point de l'écran dont les coordonnées sont indiquées entre parenthèses.

Si POINT (X, Y) vaut Ø, le point de coordonnées X et Y n'est pas allumé.

POKE

Instruction affectant un octet dans une adresse mémoire.
POKE 163Ø5,58 range le nombre 58 à l'adresse 163Ø5.
Le stockage n'est effectif que si l'adresse indiquée correspond à un octet de mémoire vive.

POS

Donne la position horizontale du curseur (colonne entre 0 et 39).
Après l'instruction LOCATE 8,10, POS(0) vaut 8.
La valeur numérique entre parenthèses derrière POS est fictive.

PRESET

Efface un point de l'écran graphique.
PRESET (100,50) efface le point de coordonnées (100,50).
PRESET STEP (– 10,20) efface le point décalé de 10 pixels à gauche et de 20 pixels en bas relativement à la position courante du curseur graphique.
PRESET (50,100),6 allume un point rouge en (50,100). (Identique à PSET.)
cf. PSET.

PRINT

Instruction d'affichage. Abréviation : ? .
PRINT imprime une liste de données à partir de la position courante du curseur d'écriture.
Les nombres sont suivis d'un blanc et précédés d'un blanc pour les nombres positifs (signe + omis).
PRINT seul force le passage à la ligne (équivalent de RETURN).
Le séparateur ; permet de fixer le curseur après le dernier caractère affiché.
PRINT A;B affiche le contenue de B après celui de A sans saut de ligne.
Le séparateur , joue le rôle d'un tabulateur avec un intervalle de tabulation de 14 colonnes.

PRINT A,B affiche le contenu de A à partir de la colonne 0 et celui de B à partir de la colonne 14.

PRINTTAB
Instruction d'affichage dans la ligne courante à partir de la colonne dont le numéro est donné entre parenthèses.
PRINTTAB(10) ; X affiche le nombre X à partir de la dixième colonne.

PRINTUSING
Instruction d'affichage respectant un format d'affichage fixé par le programmeur.
• "!" format d'affichage du premier caractère d'une chaîne.
PRINTUSING "!"."MSX" imprime M.
• "& &" format d'affichage de début de chaîne. Le nombre de caractères affichés correspond à la longueur du format (2+ nombre d'espaces entre les &). Si la longueur du format dépasse la longueur de la chaîne, des blancs sont affichés.
PRINTUSING "& &";"BASIC" imprime BAS
• Si la chaîne contient le caractère@, une chaîne donnée en argument s'insère à la place de .
PRINTUSING "LE@ DU@";"FILS" affiche LE FILS DU FILS
• Pour les nombres, le format peut préciser le nombre de chiffres affichés avant ou après le point décimal. Le format "# # #.# #" affichera tous les nombres avec trois chiffres avant et deux chiffres après le point décimal.
• Le signe + à gauche du format impose l'impression du signe.
PRINTUSING "+ #";2 imprime +2
• Le signe − à droite du format fait suivre les nombres négatifs du signe −.
PRINTUSING "#.# −";−3 imprime 3.0−
• Le signe "**" à gauche du format remplit les blancs avec des * et rajoute deux positions d'affichage.
PRINTUSING "** # #.#";7 imprime ***7.0
• Une virgule à gauche du point décimal imprime une virgule par tranche de trois chiffres dans la partie entière.
PRINTUSING "# # # #,.#";1234.74 imprime 1,234.7
• Une virgule à droite du format imprime une virgule à droite du nombre.

PRINTUSING "# #. # #,";12.3 imprime 12.30,
• Le signe "^^^^" à droite du format force l'affichage en notation scientifique.
PRINTUSING "#. #^^^^";12.3 imprime 1.2 E+01.

PRINT #
Instruction d'écriture dans un fichier suivant des règles identiques à celles de PRINT.
PRINT # permet en particulier d'imprimer du texte dans l'écran graphique. Il faut d'abord ouvrir un fichier sur l'écran par OPEN "GRP:" FOR OUTPUT AS # 1 puis d'utiliser les ordres d'affichage comme sur l'écran texte en remplaçant PRINT par PRINT # 1,.
PRINTUSING prend alors la forme : PRINT # 1,USING

PSET
Instruction d'allumage d'un point de l'écran (pixel). PSET doit être suivi des coordonnées du point choisi, et éventuellement d'un code couleur.
Le coordonnées doivent être comprises entre 0 et 255 pour la première (horizontale), entre 0 et 191 pour la deuxième (verticale).
• PSET (150,100),1 allume le point de coordonnées (150,100) en noir.
• PSET STEP (– 10,20),2 allume un point vert à 10 pixels à gauche et 20 en bas relativement à la position courante du curseur graphique.

PUT SPRITE
Instruction d'affichage d'un sprite sur l'écran graphique.
PUT SPRITE doit être suivi d'arguments correspondant à :
— le numéro du plan dans lequel la sprite sera affiché. Compris entre 0 et 31. Un seul sprite par plan.
— la position du sprite dans le plan (coordonnées graphiques). Possibilité d'utiliser STEP.
— la couleur
— le numéro du sprite

• PUT SPRITE 1,(100,20),6,1
affiche dans le plan 1, au point de coordonnées (100,20) (coin supérieur gauche du sprite), en rouge (code 6), le sprite numéro 1.

• La position, la couleur et le numéro du sprite peuvent être omis. Dans ce cas, l'affichage se fait à la position courante du curseur graphique et dans la couleur courante. Le numéro du sprite est le numéro de plan (obligatoire).
• Les plans se superposent dans l'ordre de la numérotation. Le sprite du plan 1 passera dessus le sprite du plan 2 en cas de superposition.
cf. SPRITE$, ON SPRITE, SPRITE ON/OFF/STOP

READ
Instruction d'affectation dans la variable qui suit READ, d'une constante lue dans une ligne de données (DATA).
cf. DATA, RESTORE.

REM
Instruction sans effet sur le déroulement du programme et permettant d'insérer dans le programme, des lignes de commentaires.
Abréviation : ' (apostrophe).

RENUM
Commande de renumérotation de lignes. RENUM doit être suivi du nouveau numéro, de l'ancien numéro et d'un pas.
• RENUM 100,50,20 renumérote toutes les lignes du programme au delà de 50 en commençant par 100 (l'ancienne ligne 50 devient 100) et de 20 en 20 (la première ligne suivant 50 devient 120).
• Par défaut le nouveau numéro est 10, l'ancien numéro correspond au numéro de la 1e ligne et le pas est 10. RENUM seul donne les lignes 10, 20, 30, etc.

RESTORE
Instruction plaçant le pointeur de données sur la première ligne de DATA suivant le numéro de ligne indiqué derrière RESTORE.
• RESTORE 100 place le pointeur de données sur la ligne 100 si c'est une ligne de DATA ou sur la première ligne de DATA qui suit la ligne 100.
• RESTORE seul place le pointeur de données sur la première ligne de DATA du programme.

• Cette instruction permet de lire plusieurs fois les mêmes données.
cf. DATA, READ.

RESUME
Relance le programme en cas d'erreur et exécution du branchement ON ERROR.
• RESUME 100 relance le programme à partir de la ligne 100.
• RESUME NEXT relance le programme à partir de la première ligne qui suit celle où l'erreur s'est produite.
• RESUME seul relance le programme à partir de la ligne où l'erreur s'est produite.
cf. ERROR, ERR, ERL, ON ERROR.

RETURN
Instruction indiquant la fin d'un sous-programme.
Elle commande donc le retour à la première instruction suivant l'instruction d'appel du sous-programme (GOSUB).

RIGHT$
Fonction donnant la partie droite d'une chaîne de caractères. RIGHT$ (X$, I) renvoie les I derniers caractères de X$. Si I est nul, la chaîne renvoyée est vide. Si I dépasse la longueur de X$, X$ est renvoyé.

```
RIGHT$ (''ABCD'',3)=''BCD''
RIGHT$ (''ABCD'',0)=''''
RIGHT$ (''ABCD'',5)=''ABCD''
```

RND
Donne un nombre aléatoire.
• RND(1) donne un nombre entre 0 et 1.
• RND(0) donne toujours le nombre précédent obtenu par RND(0).
• RND(N) donne un nombre aléatoire dépendant de N si N est négatif.

Remarque : pour obtenir des tirages différents à chaque déroulement, on utilisera la routine suivante qui simule un RANDOMIZE en testant le clavier.

```
100 X=RND(1) : IF INKEY$=''''THEN 100
```

RUN
Commande d'exécution du programme présent en mémoire vive.
Avant de lancer l'exécution, toutes les variables sont remises à 0 et le pointeur de données est placé sur la première ligne de DATA.
• RUN 100 lance l'exécution à partir de la ligne 100.
(GOTO 100 en commande a le même effet sans réinitialisation des variables.)

SAVE
Sauvegarde le programme sur le lecteur de cassettes en mode ASCII.
• SAVE "CAS:TRUC" enregistre le programme contenu en mémoire vive sur la cassette et sous le nom TRUC qui permettra de le rappeler.
• La sauvegarde en mode ASCII permet le fusionnement de deux programmes en mémoire vive (cf. MERGE).

SCREEN
Instruction permettant de sélectionner l'écran, la taille des sprites, la sonorisation du clavier, la vitesse de transmission des données avec le lecteur de cassettes, et le type d'imprimante (arguments numériques placés derrière SCREEN dans cet ordre).
• L'écran : sélection de l'un des 4 types d'affichage.
0 : Mode texte. 40 colonnes × 24 lignes.
1 : Mode texte. 32 colonnes × 24 lignes.
2 : Mode graphique haute résolution : 256 colonne × 192 lignes.
3 : Mode graphique basse résolution : 64 colonnes × 48 lignes.
Par défaut, le mode est 0.
• La taille des sprites : sélection de l'une des 4 tailles disponibles pour les sprites.
0 : sprite défini dans un pavé 8 × 8 affiché en simple taille.
1 : sprite défini dans un pavé 8 × 8 affiché en double taille.
2 : sprite défini dans un pavé 16 × 16 affiché en simple taille.
3 : sprite défini dans un pavé 16 × 16 affiché en double taille.
Par défaut, la taille est 0.
• La vitesse de transmission : sélection de l'une des 2 vitesses disponibles pour le lecteur de cassette.
1 : 1200 bauds
2 : 2400 bauds
Par défaut, la vitesse est 1.

• Le type d'imprimante
0 : pour une imprimante dédié.
1 : pour une imprimante quelconque compatible.
Par défaut, le type est 0.
• Exemple : SCREEN 2,1,1,1
sélectionne l'écran graphique haute résolution, des sprites 8 × 8 en double taille, une vitesse de transmission avec le lecteur de cassettes à 1200 bauds et une imprimante compatible.

SGN

Fonction donnant le signe du nombre placé entre parenthèses et suivant le code : 1 pour les positifs, -1 pour les négatifs et Ø pour Ø.

```
SGN (5) = 1
SGN (0) = 0
SGN (-4) = -1
```

SIN

Donne le sinus d'un angle exprimé en radians.

```
SIN (0) = 0
SIN (3.14) = 0
SIN (1.5) = .9974949498660406
```

SOUND

Instruction donnant accès aux registres du circuit son. SOUND est suivi du numéro de registre choisi (entre 0 et 13) et d'une valeur numérique entre 0 et 255. Les valeurs numériques associées à chacun des registres permettent de régler la fréquence, le volume, l'enveloppe et la période de l'enveloppe pour les 3 générateurs de son (A,B,C) et le générateur de bruit.
• Fréquence
La fréquence du générateur A se règle par l'intermédiaire des canaux 0 et 1. Elle est codée sur 2 octets : le poids fort pour le canal 0, le poids faible pour le canal 1.
SOUND 0,X : SOUND 1,Y produit sur le générateur A un son de fréquence 256*X + Y.
De même la fréquence du générateur B se règle sur les canaux 2, 3 ;

celle du générateur C sur les canaux 4 et 5. La fréquence est entre 0 et 65535 (entre 0 et 4080 Hertz).

• Volume

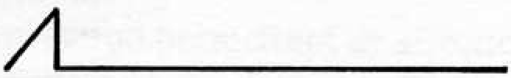
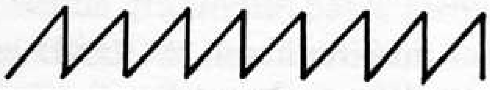
Le volume du générateur A se règle sur le canal 8 (canal 9 pour B et 10 pour C). La valeur du volume doit être comprise entre 0 et 15. Par défaut, c'est 8.

SOUND 9,15 règle le volume du canal B au maximum.

Si le volume est réglé à 16, le volume de sortie du générateur concerné est contrôlé par le générateur d'enveloppe.

• Enveloppe

Les enveloppes communes des 3 générateurs se fixent sur le seul canal 13. 8 enveloppes sont accessibles en choisissant d'affecter au canal 13 un entier entre 0 et 15 (deux enveloppes sont accessibles par plusieurs nombres).

Enveloppes	valeur affectée au canal 13
	0 ; 1 ; 2 ; 3 ; 9
	4 ; 5 ; 6 ; 7 ; 15
	8
	10
	11
	12
	13
	14

SOUND 13,10 donne une enveloppe triangulaire pour les trois générateurs.

• Période de l'enveloppe

La période commune des trois générateurs se règle sur les canaux 11 et 12 (2 octets). Elle est codée sur 2 octets : le poids fort pour le canal 11, le poids faible pour le canal 12.

SOUND 11,X : SOUND 12,Y règle l'enveloppe à 256*X+Y.

• Générateur de bruit

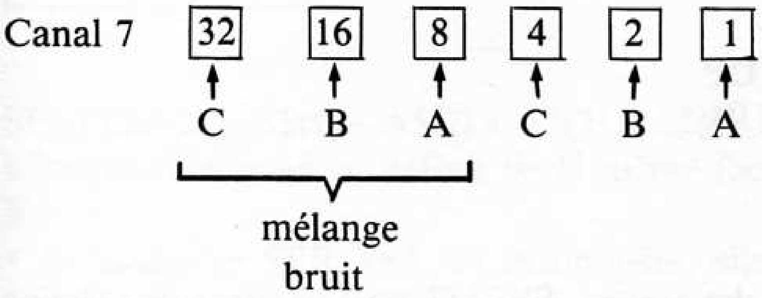
Le niveau du générateur de bruit se règle sur le canal 6 grâce à une valeur comprise entre 0 et 63.

SOUND 6,50 règle le niveau du générateur de bruit sur 50.

SOUND 6,0 supprime l'émission de bruit.

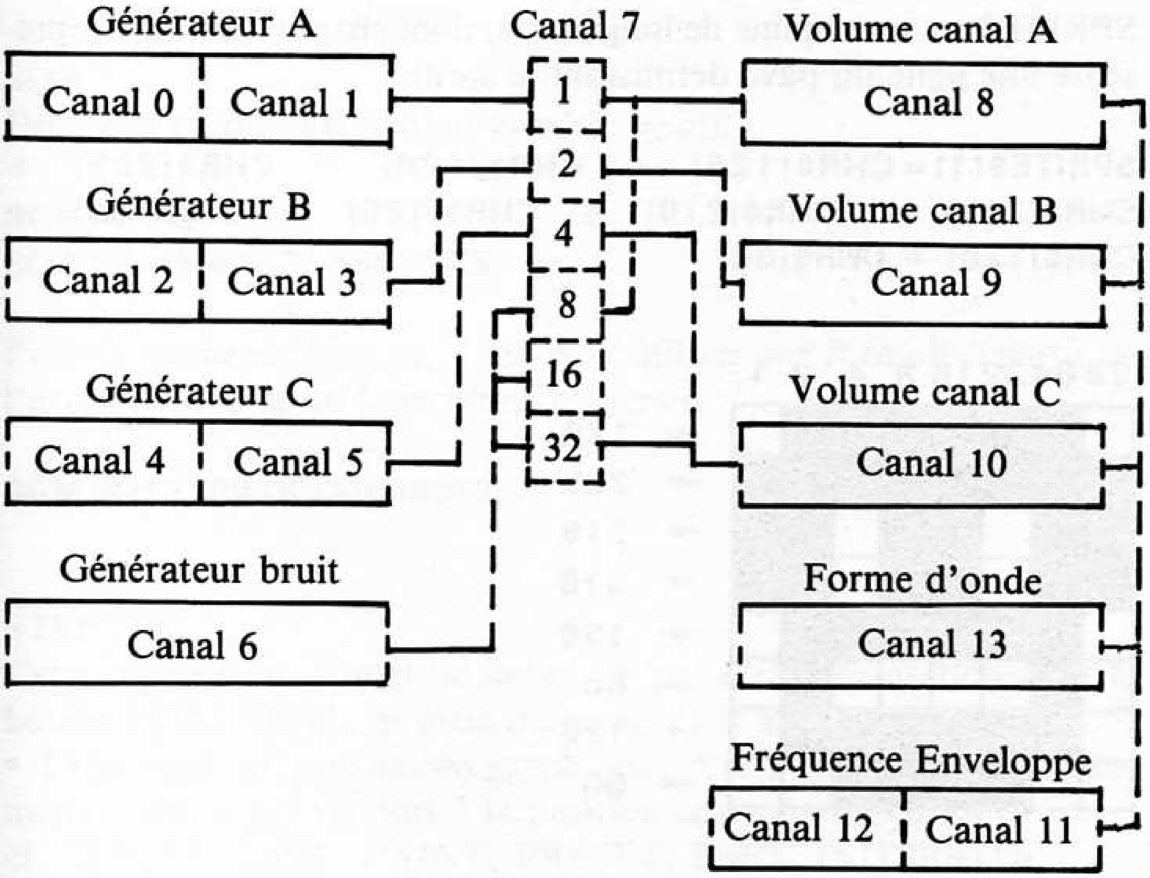
• Le canal 7

Le canal 7 est une voie de passage pour les trois générateurs A, B et C et le mélange du bruit sur ces générateurs.



Le passage est effectif lorsque le bit correspondant est à 0.

Schéma résumant le rôle respectif des 14 canaux



SPACE$

Donne une chaîne ne contenant que des espaces.

- SPACE$ (4) est égal à la chaîne “   ” (4 blancs).
- L'argument doit être compris entre 0 et 255.

SPC

Imprime des espaces à l'écran ou sur l'imprimante.

- PRINT SPC(4) imprime 4 espaces.
- L'argument doit être compris entre 0 et 255.

SPRITE ON/OFF/STOP

cf. ON SPRITE GOSUB.

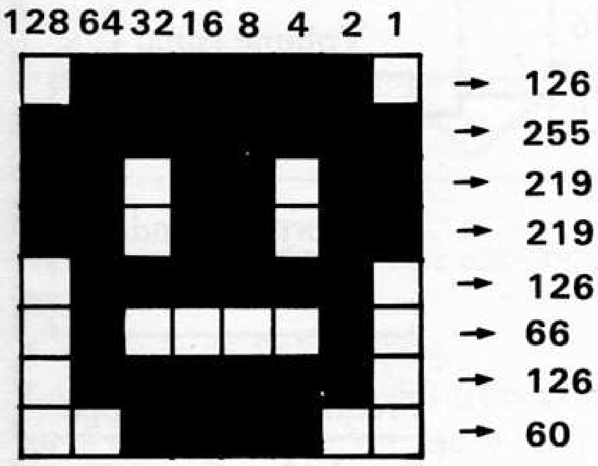
SPRITE$

Variable de définition des sprites. SPRITE$(N) contient une chaîne décrivant le sprite numéro N. Le nombre maximum de sprites définissables dépend de la taille de ceux-ci : 256 en taille normale, 64 en taille double.

- SPRITE 8 × 8 (taille normale)

SPRITE$ est une chaîne de longueur 8, dont chaque caractère représente une ligne du pavé définissant le sprite.

```
SPRITE$(1)=CHR$(126) + CHR$(126) + CHR$(255) +
CHR$(219) + CHR$(219) + CHR$(126) + CHR$(66) +
CHR$(126) + CHR$(60)
```



• SPRITE 16 × 16 (taille double)
SPRITE$ est une chaîne de longueur 32, décomposée en 4 chaînes de 8 caractères représentant 4 pavés 8 × 8.

A$ (1)	A$ (3)
A$ (2)	A$ (4)

SPRITE$(2) = A$(1) + A$(2) + A$(3) + A$(4)
Chaque chaîne A$ se définit de la même façon que pour un sprite 8 × 8.
• Si la chaîne SPRITE$ est incomplète, elle est automatiquement complétée par des CHR$(0).
• La taille (8 × 8 ou 16 × 16) se fixe par l'instruction SCREEN pour chacun des 32 plans graphiques.
cf. ON SPRITE GOSUB, SCREEN.

SQR
Donne la racine carrée d'un nombre positif.

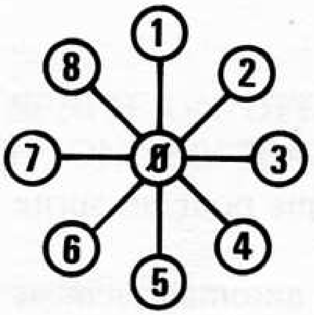
```
SQR (4) = 2
SQR (2) = 1.414213562373
```

Pour la racine cubique de X, on peut utiliser une formule à partir de l'exponentielle et du logarithme méperien :

```
SGN (X)*EXP(LOG(ABS(X))/3)
```

STEP
Cette instruction permet de définir le pas d'incrémentation dans la boucle FOR, NEXT. Si cette instruction est absente, le pas est 1.
• Dans cetaines instructions graphiques, STEP indique des déplacements relatifs par rapport à la position courante du curseur.
cf. CIRCLE, LINE, PAINT, PRESET, PSET, PUTSPRITE.

STICK
Donne la direction du manche de la manette de jeu.
• STICK(0) concerne les touches de déplacements du curseur considérées comme une manette de jeu. L'appui simultané de deux touches voisines provoquant des déplacements diagonaux.
↑ + ↘ équivaut à ↗
• STICK(1) concerne l'interface joystick 1.
• STICK(2) concerne l'interface joystick 2.
• Le codage des 9 directions permises est le suivant.



STOP
• Marque un point d'arrêt dans le programme.
Le numéro de la dernière ligne exécutée s'affiche. Les variables sont inchangées. L'utilisateur peut relancer le déroulement par CONT.
• Voir ON STOP GOSUB pour les instructions STOP ON, STOP OFF et STOP STOP.

STRIG
Donne l'état de la barre du clavier ou du bouton de chaque manette de jeu.
• STRIG(0) concerne la barre du clavier
• STRIG(1) et STRIG(3) concernent le joystick 1.
• STRIG(2) et STRIG(4) concernent le joystick 2.
Si la touche ou le bouton est appuyé la valeur renvoyée est (−1) et 0 dans le cas contraire.

STRIG ON/OFF/STOP
• STRIG ON active l'instruction de branchement ON STRIG GOSUB
• STRIG OFF la désactive
• STRIG STOP l'inhibe.

STRIG$

Donne une chaîne de caractères dépendant des arguments fournis.
• STRING$(N,A$) renvoie une chaîne contenant N fois le premier caractère de A$.
STRING$ (3,"BAS") donne "BBB"
• STRING$ (N,X) renvoie le caractère de code ASCII X.
N est fictif
STRING$ (5,65) donne "A"

STR$

Fonction de conversion d'une expression numérique en une chaîne de caractères (voir VAL).

SWAP

Instruction permettant d'échanger les contenus de deux variables de même type : SWAP A, B échange les contenus des variables numériques A et B.

TAB

cf. PRINTTAB.

TAN

Donne la tangente d'un angle exprimé en radians.

TAN (0) = 0
TAN (1.5) = 14.10141994718

THEN

cf. IF,THEN,ELSE.

TIME

Donne le temps écoulé depuis la mise sous tension. Le compteur TIME est incrémenté d'une unité toutes les 60 secondes. TIME est conservé pendant les interruptions (lecteur de cassette par exemple).

TROFF
Commande d'annulation de la commande TRON.

TRON
Commande utilisée pour la mise au point d'un programme. Lors de l'exécution, les numéros de lignes sont affichés entre crochets dans l'ordre de leur exécution.
cf. TROFF.

USR
Appel d'une routine en langage machine.
USR doit être suivi du numéro de la routine (entre 0 et 9) et de l'argument transmis.
USR2,A transmet à la routine 2 l'argument numérique contenu dans A. L'adresse de lancement doit être préalablement définie par DEFUSR.

VAL
Renvoie la valeur numérique d'une chaîne de caractères, c'est-à-dire le premier nombre rencontré dans la chaîne si la chaîne commence par un chiffre.

```
VAL ("18")=18
VAL ("18 FRANCS")=18
VAL ("A54")=0
```

VARPTR
Fonctions donnant l'adresse du premier octet où le contenu de la variable indiquée entre parenthèses est rangé.
- Les variables entières sont rangées dans 2 octets consécutifs.
- Les variables réelles simple précision sont rangées dans 4 octets consécutifs.
- Les variables réelles double précision sont rangées dans 8 octets consécutifs.
- Les chaînes de caractères sont précédées d'un descripteur codé sur 3 octets (VARPTR renvoie l'adresse du premier de ces 3 octets) : le 1e

octet du descripteur contient la longueur de la chaîne, les deux suivants, l'adresse où la chaîne est physiquement stockée.

VDP
Instruction de lecture des 8 registres du circuit vidéo et du microprocesseur.
- VDP(0) à VDP(7) concerne le circuit vidéo
- VDP(8) concerne le microprocesseur (statut)

VPEEK
Instruction de lecture dans la mémoire écran (adresses 0 à 16383) haute résolution.
PRINT VPEEK (1000) renvoie l'octet affiché dans la mémoire écran à l'adresse 1000.
cf. VPOKE.

VPOKE
Instruction de lecture dans la mémoire écran haute résolution (adresses 0 à 16383).
VPOKE 1000,150 range la valeur 150 à l'adresse 1000 de la mémoire écran.

WAIT
Attente d'un octet sur l'un des ports d'entrée/sortie du microprocesseur.
- WAIT 1,64 interrompt le déroulement du programme jusqu'à apparition de l'octet 64 sur le port numéro 1.

WIDTH
Modification de la largeur de l'écran texte. Le nombre de colonnes dépend de l'option SCREEN courante (32 ou 40).
- WIDTH 25 limite l'affichage texte dans les 25 première colonnes.

XOR
Ou exclusif logique. Placée entre deux expressions numériques, cette

opération donne une valeur correspondant à vrai ou faux suivant la table d'opération :

XOR	Ø	1
Ø	Ø	1
1	1	Ø

Ø : Faux
1 : Vrai

Chez le même éditeur

L'informatique, premier contact, *Jean-Pierre Petit.*
Lire LSE, *Christian Lafond/Pierre Muller.*
Lire BASIC, *François-Marie Blondel/Jean-Claude Le Touze.*
Lire PASCAL, *Dominique Piot.*
Dictionnaire de micro-informatique et de micro-électronique, *Christiane Morvan.*
Premier livre de programmation, *Odette Arsac-Mondou/Christiane Bourgeois-Camescasse/Mireille Gourtay.*
Deuxième livre de programmation, *Odette Arsac-Mondou/Christiane Bourgeois-Camescasse/Mireille Gourtay.*
Premières leçons de programmation, *Jacques Arsac.*
L'algorithmique de la pratique à la théorie, *Guy Charty/Jean Vicard.*
Cobol vol. I, *Pierre Bouchet/Jean Vicard.*
Les microprocesseurs, *François-Marie Blondel/Wladimir Mercouroff.*
Une expérience d'EAO, *Marie-Françoise Gibert-Doutre.*
Enseignement et Ordinateur, *Hélène Bestougeff/Jean-Pierre Fargette.*
Les ordinateurs, *Wladimir Mercouroff.*
S'organiser avant de s'informatiser, *Bernard Espaze.*
Le traitement de texte, point de départ de la bureautique, *Jean-Michel Trouche.*
L'anglais pour informaticien, *Liliane Gallet.*
Sachet choisir votre micro-ordinateur de gestion, *Beranger Le Breton.*
Initiation au BASIC TO7/TO7-70, *Christine et François-Marie Blondel*
Le BASIC D.O.S. du TO7/TO7.70 et du MO5, *Christine et François-Marie Blondel*
Un ordinateur à la maison, *Jean Delcourt*
Un ordinateur en fête, *Serge Pouts-Lajus*
Un ordinateur et des jeux, *Jean-Pascal Duclos*
Guide pratique de l'ordinateur personnel d'IBM, *Dalloz/Emery/Portefaix/Boisgontier/Salzman*
LOGO, des ailes pour l'esprit, *Horacio C. Reggini*
Premiers pas avec le ZX-SPECTRUM, *Ian Stewart/Robin Jones*
Le langage machine du ZX-SPECTRUM, *Ian Stewart/Robin Jones*
Plus loin avec le ZX-SPECTRUM, *Ian Stewart/Robin Jones*
Jeux vidéo, jeux de demain, *Georges-Marie Becherraz/Alain Graber*
Guide pratique de l'Oric-Atmos, du Basic à l'assembleur, *Michel Bussac/Robert Lagoutte*
Des programmes pour votre Oric-Atmos, *Michel Piot*
Premiers pas avec le Commodore 64, *Ian Stewart/Robin Jones*
Le guide du MO5, *André Deledicq*
Guide pratique de l'enseignement assisté par ordinateur, *Jean-Michel Lefèvre*
Faites vos jeux en assembleur sur TO7/TO7-70, *Michel Oury*
Initiation à Logo, *Doris Avram/Michèle Weidenfeld/l'équipe de S.O.L.I.*
Logo, manuel de référence, *D. Avram/T. Savatier/M. Weidenfeld/S.O.L.I.*
Initiation au Forth, *S.E.F.I.*
Forth, manuel de référence
Manuel de l'assembleur 6809 du TO7/TO7-70, *Michel Weissgerber*
Manuel de l'assembleur 6809 du MO5, *Michel Weissgerber*
La face cachée du TO7/TO7-70, *Jean-Baptiste Touchard*

Manuel technique du TO7/TO7-70, *Michel Oury*
Manuel technique du MO5, *Michel Oury*
Macintosh, Multiplan, Macpaint, *Eddie Adamis*
Vous et l'ordinateur APPLE, *Edward H. Carlson*
Écrivons un programme pour APPLE, *Royal Van Horn*
Le Logo sur APPLE, *Harold Abelson*
Musique sur COMMODORE 64, *James Vogel/Nevin B. Scrimshaw*
Guide pratique du VIDÉOTEX et du MINITEL, *Saboureau/Bouché*
Guide pratique de l'enseignement assisté par ordinateur, *Jean-Michel Lefèvre*

Imprimé par Pollina, 85400 Luçon - N° 7014
Dépôt légal : Avril 1985
Imprimé en France

Programmes pour MSX

Le Basic MSX est un standard qui offre de grandes possibilités pour le calcul, le graphique et la musique sur micro-ordinateur.
Vingt programmes commentés invitent ici à en tirer le meilleur parti.
Cet ouvrage vous propose également un dictionnaire des mots clés du Basic MSX pour comprendre et créer vous-même des programmes spectaculaires et originaux.

ISBN : 2-7124-**0582**-X F 75/85/06